



Running Multiple Simulation Trials in OneSAF

by MaryAnne Fields and MyVan Baranoski

ARL-TR-3322

September 2004

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5066

ARL-TR-3322

September 2004

Running Multiple Simulation Trials in OneSAF

MaryAnne Fields and MyVan Baranoski
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE				<i>Form Approved</i> OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) September 2004		2. REPORT TYPE Final		3. DATES COVERED (From - To) October 2002–June 2004	
4. TITLE AND SUBTITLE Running Multiple Simulation Trials in OneSAF				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) MaryAnne Fields and MyVan Baranoski				5d. PROJECT NUMBER 622618.AH80	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRD-ARL-WM-BF Aberdeen Proving Ground, MD 21005-5066				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-3322	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT In this report, we describe a method to run multiple simulation trials efficiently using OneSAF. Our discussion focused on three main areas: managing the experimental trials, specifying experimental parameters with a graphical user interface, and collecting experimental data. The work presented in this report describes a methodology rather than a simulation tool—users will have to tailor the methodology to their individual simulation studies.					
15. SUBJECT TERMS simulation, modeling, iteration, OneSAF					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UL	18. NUMBER OF PAGES 64	19a. NAME OF RESPONSIBLE PERSON MaryAnne Fields
a. REPORT UNCLASSIFIED	b. ABSTRACT UNCLASSIFIED	c. THIS PAGE UNCLASSIFIED			19b. TELEPHONE NUMBER (Include area code) 410-278-6675

Contents

List of Figures	iv
1. Introduction	1
2. Managing a Series of Experimental Trials	2
2.1 Creating Units.....	2
2.2 Assigning Missions	3
2.3 Managing the Trials.....	5
3. The User Interface	7
4. Collecting Data	11
5. Examples	11
5.1 Simple Meeting Engagement Involving Two Vehicles	11
5.2 Experiment II Overwatching Fires (OWF) Behavior.....	11
6. Conclusions	13
7. References	15
Appendix A. The UUTIL_UNIT_PARAMS Structure	17
Appendix B. The M1 Experiment Editor	19
Appendix C. The Reader File for Editor Shown in Figure 9	21
Appendix D. The Overwatching Fires Experiment Editor	23
Distribution List	52

List of Figures

Figure 1. M1_Experiment function – variable definitions and section 1.	3
Figure 2. M1_Experiment function – section 2.	4
Figure 3. M1_Experiment function – section 3.	4
Figure 4. M1_Experiment function – section 4.	4
Figure 5. M1_Experiment function – section 5.	5
Figure 6. A simple experiment manager function for the M1 experiment – section 1.	6
Figure 7. A simple experiment manager for the M1 experiment function – section 2.	7
Figure 8. A simple experiment manager for the M1 experiment function – section 3.	7
Figure 9. libexp.rdr file – section 1.	8
Figure 10. libexp.rdr file – section 2.	8
Figure 11. libexp.rdr file – section 3.	9
Figure 12. The exp_edt_struct structure from the libexp.h file.	9
Figure 13. GUI editor for simple meeting engagement experiment.	10
Figure 14. The OWF experiment editor.	13

1. Introduction

In our work with weapon system concepts, we frequently need to parameterize battlefield scenarios by varying characteristics (e.g., ammunition load, response time, speed, location, etc.) of the weapons systems in the simulation experiment. Depending on the number of weapon system characteristics being studied, a simulation experiment may require ten to hundreds of repetitions of individual experimental trials. For some experiments, we can write a specialized self-contained simulation that loops through all the variations we wish to study. For other experiments, we need to use the OneSAF battlefield simulation tool. OneSAF is a real-time distributed interactive simulation tool developed by the U.S. Army Simulation, Training, and Instrumentation Command for training and analysis. It can be used to examine weapons systems concepts in scenarios as large as brigade-level. It can also be used to model engagements as small as one-on-one encounters. To run an exercise, a user places friendly and opposing forces on a digital terrain database, assigns missions for each unit in each of the forces in the scenario, and runs the simulation. This process can be made somewhat more efficient by saving and reloading scenario-related parameters such as initial unit locations and planned movement routes. However, running several variations for a parametric study still requires considerable user involvement.

In this report, we describe a method to run multiple simulation trials efficiently using OneSAF. Our discussion focuses on three main areas: (1) managing the experimental trials, (2) specifying experimental parameters with a graphical user interface (GUI), and (3) collecting experimental data. The work presented in this report describes a methodology rather than a simulation tool—users will have to tailor the methodology to their individual simulation studies.

We illustrate our methodology with two experiments. In the first experiment, a single unmanned ground vehicle (UGV) travels through rough terrain. In the second experiment, a team of UGVs patrols a road segment on the battlefield.

This document refers to code libraries, functions, and structures contained in the OneSAF simulation code. To avoid confusion, we adopt the following notation: library names are printed in bold typeface (i.e., **libctdb**), function names are italicized (i.e., *unitdefdb_name_to_object*), and C structure names are capitalized (i.e., UUTIL_UNIT_PARAMS). We frequently illustrate the text with code examples. Large, multipage programs will appear in the appendices (appendices A–D). Small code samples will appear as figures in the main text; these samples will be indented and boxed to distinguish them from the main discussion.

To help users locate files within the OneSAF code, we will assume that the root directory of OneSAF is “OneSAF_Root”; all file names include directory paths relative to this root directory. File names are offset from the main text by quotes.

2. Managing a Series of Experimental Trials

In this section, we describe a method to design and control a simulation experiment with minimal user involvement. Because our methods involve writing application libraries for the OneSAF simulation tool, the reader should be familiar enough with OneSAF and the C programming language to modify and recompile the code. Our work utilizes the following OneSAF libraries: **libsched** (1), **libtaskutil** (2), **libunitutil** (3), **libpo** (4), **libentity** (5), **libtime** (6), **libunitdefdb** (7), **libctdb** (8), **libcoordinates** (9), and **libeditor** (10). It is beyond the scope of this report to include detailed discussions of these libraries; however, we include a brief description of the specific library functions used in our work.

We begin our discussion by describing the software necessary to simulate a set of experimental trials. Similar to any OneSAF simulation experiment, in our work, we need to place units on the battlefield, assign missions, and manage the set of experimental trials. However, because user involvement for our work is restricted to using an editor to set up parameters for a series of experimental runs, these setup tasks must be handled automatically by the software. The experiment tool we developed utilizes existing OneSAF software functions to control the placement and function of units in a scenario.

2.1 Creating Units

Units are created using functions and data structures from the **libunitutil** library. The unit characteristics are specified by the `UTIL_UNIT_PARAMS` structure shown in appendix A. In an application, a user creates a `UTIL_UNIT_PARAMS` variable and assigns values to its parameters. In this discussion, we concentrate on the structure variables shown in bold italics—*object_type*, *location*, *direction*, and *formation*. The unit is specified by the *object_type* variable of the structure. Generally, a unit name, such as “unit_US_M1_Platoon” is translated into its object type using the function *unitdefdb_name_to_object*, from the **libunitdefdb** library, and the unit definition database for the simulation.

The *location* variable specifies the initial location in meters (X, Y, and Z) for a unit in a geocentric coordinate system (11) along with a terrain cell C. For some experiments, the initial location of the units is constant across all the experimental trials. The user may wish to specify this initial location in a more familiar coordinate system such as Universal Transverse Mercator (UTM) and use the **libcoordinates** library to translation to UTM. Other experiments may require the initial location to vary within a user-specified region. Initial locations may also depend on scenario requirements such as the use of roads or cover. The initial *direction* can be constant for all experimental trials or it can be calculated from the positions of opposing units or the location of waypoints or other points of interest.

The formation parameter is a character string describing the military formation or pattern in which a unit moves. Common values for the formation parameter are *line*, *wedge*, and *vee*. The library **libformationdb** contains available formations along with tools to create and modify formations. In this work, the formation is fixed for all the experimental trials.

Once the unit parameter structure has been initialized, the function *unitutil_create* creates an entry in the persistent object database for that unit. The persistent object database manages all the objects, such as units, waypoints, overlays, and routes, in the simulation.

2.2 Assigning Missions

Many simple missions in OneSAF are described by a single taskframe. A taskframe is a collection of related tasks that run simultaneously. In general, a taskframe includes a preparatory phase and a primary task. The preparatory phase allows the unit to perform any initialization activities before executing the main task. Usually, it is set to a Halt task. Taskframes may also include reaction tasks that allow units to interrupt the primary task to respond to situations such as enemy fire. Definitions for existing taskframes are found in the file “OneSAF_Root/src/onesaf/taskframe.rdr”. The **libtaskutil** library contains functions to assign and configure tasks. The function *taskutil_create_frame* creates an instance of a specific taskframe. This function must use an existing taskframe from “OneSAF_Root/src/onesaf/taskframe.rdr”. Users can set parameters for the taskframe using the *taskutil_set_parameter* function.

Figures 1–5 show a C function, *M1_Experiment*, which places an M1 platoon at an arbitrary point on the battlefield and assigns it a move mission. In this discussion, the function is divided into sections to improve readability. Appendix A lists the complete C function. The first section defines the local variables used in the *M1_Experiment* function. This section also translates the unit name into its corresponding object.

```
static void M1_Experiment (PO_DATABASE *db)
{
    /* Variable definition */
    PO_DB_ENTRY *frame, *point_entry, M1Unit_entry;
    int32 vehicle_id, methodology, sx, sy, ex, ey;
    float64 speed, angle;
    uint32 size;
    char      *otype, unit_type [80], formation [80];
    ObjectType      object;
    ObjectID        M1Unit_ID;
    UnitClass       M1Unit;
    UUTIL_UNIT_PARAMS new_M1Unit;
    PO_BUFFER (PointClass, point, pointbuf);

    /*1. Find the object from the unit name and the unit_db*/
    otype = stdname_get_standard (reader_get_symbol ("unit_US_M1_Platoon"));
    object = unitdefdb_name_to_object_type (otype, unit_db);
}
```

Figure 1. M1_Experiment function – variable definitions and section 1.

```

/*2. Set up the unit parameters structure */
bzero (&new_ M1Unit, sizeof (UTIL_UNIT_PARAMS));
M1Unit.force = distinguishedForceID;
M1Unit.methodology = 0;
M1Unit.mounted = FALSE;
strcpy ((char *) formation, "line");
M1Unit.formation = reader_get_symbol (formation);
M1Unit.object_type = object;
sx = 5000.0 + 1000.0*drand48 ();
sy = 5000.0 + 1000.0*drand48 ();
ex = sx + 500.0*drand48();
ey = sy + 500.0*drand48 ();
M1Unit.location[0] = sx;
M1Unit.location[1] = sy;
angle = atan((eby - sby)/(ebx - sbx));
M1Unit.direction = RAD_TO_BAM(angle);
M1Unit.competence = 1.0;
M1Unit.use_default_munitions = TRUE;
M1Unit.sub_formation = reader_get_symbol("line");

```

Figure 2. M1_Experiment function – section 2.

```

/*3. Create the M1 platoon */
M1Unit_entry = unitutil_create(&M1Unit, NULL,0,NULL, 0, TRUE, db);
M1Unit_unit = PO_UNIT_DATA(M1Unit_entry);
overlay = po_get_object(db, &PO_UNIT_DATA(M1Unit_entry).overlayID);

```

Figure 3. M1_Experiment function – section 3.

```

/* 4. Create a point for the move mission */
bzero(&point, sizeof(point));
point.overlayID = PO_UNIT_DATA(M1Unit_entry).overlayID;
point.style = PSgeneral;
point.color = OCGreen;
point.dashed = 0;
point.location.x = ex;
point.location.y = ey;
point.direction = RAD_TO_BAM(DEG_TO_RAD((float64) degrees));
point_entry = po_create_object(db, NULL, objectClassPoint,
                             FALSE, &point, sizeof(point), NULL);

```

Figure 4. M1_Experiment function – section 4.

In figure 2, the second section sets up the unit parameters as previously discussed. The initial location, (sx, sy), is chosen so that it lies in a 1- × 1-km rectangular region of the battlefield that is 5 km from the western and southern borders of the battlefield. There is no error checking in this example; in an actual program, it is important to check that the point (sx, sy) is actually

```

/* 5. Create the Move Mission */
frame = taskutil_create_frame("Move");
taskutil_set_parameter(frame, SM_UMixedTravel, "route",
                      &PO_OBJECT_ID(point_entry));

speed = 20.0;
taskutil_set_parameter(frame, SM_UMixedTravel, "speed", &speed);
M1Unit_ID = PO_OBJECT_ID(M1Unit_entry);
taskutil_assign(frame, &M1Unit_ID);
}

```

Figure 5. M1_Experiment function – section 5.

on the battlefield. The destination point, (ex, ey) is generated at the same time as the initial point. This point lies within 0.5 km of the initial point. Both points depend on the standard C random number function, `drand48`. The vector starting at (sx, sy) and terminating at (ex, ey) gives the direction of the unit. In this section, we also initialize the driving competence, force type, and formation parameters.

The third section, figure 3, uses the M1 unit structure to create an M1 platoon. We also create an overlay for the M1 platoon to store the destination point, (ex, ey) used in the move mission. An overlay contains map graphics such as points, lines, and text. Overlays can be private so that they are only visible to specific units or public, so that they are visible to all units in the exercise. A researcher can see the overlay associated with the M1 platoon by selecting the unit from the graphical display.

The fourth section, figure 4, creates a OneSAF point structure for the destination point (ex, ey). The structure specifies the type, color, location, and orientation of the point. The point is placed on the M1 overlay.

The last section, figure 5, creates a Move taskframe for the M1 platoon. This taskframe contains the SM_MixedTravel behavior as its primary task. The `taskutil_set_parameter` function is used to set the route and speed for this behavior. The function `taskutil_assign` on the last line of the code actually starts the task for the unit—this function is equivalent manually issuing an “On Order” command to start a mission.

2.3 Managing the Trials

An interactive user can easily stop and start experimental trials by using the GUI. Our experiment tool needs an exit criterion to stop the experimental trials. There are three types of exit criteria we will consider—mission completion, trial duration, and unacceptable level of casualties. “Mission completion” is the simplest of the exit criteria—the experiment terminates when the mission is accomplished. To use this criterion, the experiment tool must monitor the task state. The trial terminates when the main task of the taskframe, described in the previous section, reaches its terminal state. This is a useful criterion for missions involving little risk of failure. We have used mission completion as the exit criteria for a series of experimental trials to

evaluate the planning phase of more complex missions. Trial duration allows each experimental trial to run for a specified amount of time. The advantage of this criteria is that it is very easy to program—it only needs to monitor the simulation clock. The disadvantage is that it may be difficult to time the trials so that the mission is completed when the trial ends. Using “casualty level” as an exit criterion is useful for trials involving opposing forces. Here, the experiment tool must monitor casualty levels using functions from the **libunitutil** library. It is possible to develop an exit criteria that depends on all three of the criteria previously discussed or on other factors such as distance between opposing forces, loss of specific assets, or the number of rounds fired.

In this example, we have created a simple experiment manager function. It runs 20 trials of the M1 platoon experiment shown in figures 1–5. Each experimental trial runs for 10 min. Platoons are destroyed at the end of each trial so that each trial starts with a “fresh” unit. The code is divided into three sections. Figure 6 shows a part of the simple experiment manager function, which manages the experimental trials. The conditional block takes one of three actions—it starts the first trial, it stops the experiment (and the program), or it switches experimental trials.

```
void exp_manager (PO_DATABASE *db)
{
    static int32 elapsed_time = 0, start_time = 0;
    static int32 tickNumber = 0, trial_count = 0;
    int32 current_time;
    /* 1. Manage current trial */
    if (tickNumber == 0) /* Initialize the manager */
    {
        start_time = time_last_simulation_clock;
        sched_deferred_fncl ((SCHED_FUNCTION) M1_Experiment, 100,
            my_group, A_PTR, db, A_END);
    }
    else if (trial_count == 20) /* Quit after 20 iterations */
    {
        exit (0);
    }
    else if (elapsed_time > 10*60*1000) /* Run each trial for 10 minutes */
    {
        unitutil_delete_object (db, &PO_OBJECT_ID (M1Unit_entry), TRUE);
        /* Remove M1 platoon */
        sched_deferred_fncl ((SCHED_FUNCTION) M1_Experiment, 100,
            my_group, A_PTR, db, A_END); /* Schedule next trial */
        start_time = time_last_simulation_clock; /* Reset the clock */
        elapsed_time = 0;
        trial_count = trial_count + 1;
    }
}
```

Figure 6. A simple experiment manager function for the M1 experiment – section 1.

The second section, figure 7, keeps track of the elapsed time for each trial.

```
/* 2. Collect time information*/
tickNumber = tickNumber + 1;
current_time = time_last_simulation_clock; /* Get current time */
elapsed_time = current_time - start_time; /* Compute elapsed time */
```

Figure 7. A simple experiment manager for the M1 experiment function – section 2.

Finally, the third section reschedules the *exp_manager* function (figure 8). The function *sched_deferred_fncl* is found in the **libschedule** library. It is one of several functions used to manage reoccurring events in the simulation.

```
/* 3. Reschedule the manager */
sched_deferred_fncl ((SCHED_FUNCTION) exp_manager, 100,
my_group, A_PTR, db, A_END);
}
```

Figure 8. A simple experiment manager for the M1 experiment function – section 3.

3. The User Interface

User involvement is restricted to using a GUI editor to set up parameters for a sequence of experimental trials. OneSAF provides very flexible utilities for designing user interfaces. These utilities are described in **libeditor** and examples of editors are provided in many of the OneSAF libraries. We used HHour (**libhhour**) editor and the units (**libunits**) editors as templates for the experiment editor. The EDITOR structure, typically defined in the “libexp.h” file, and the reader file, “libexp.rdr,” determine the design of the editor. The EDITOR structure defines the variables required by the editor.

The EDITOR structure is defined in two places—the structure shown in figure 9 from the “libexp.h” file, and in the reader file, “libexp.rdr” (figures 9–11 and appendix B). The first section, shown in figure 9, defines the EDITOR structure. The variables in this section are the same as those contained in the EDITOR structure shown in figure 12. OneSAF calculates the memory for each editor based on the maximum number of bytes necessary for the structure. Consequently, the order and size of the variables must be the same in both structures. Mixing data types such as 64-bit floating point numbers, 32-bit integer numbers, and 8-bit character data may require the user to add padding variables to ensure that the variables start at the beginning of

;;Editor definition for the Experiment editor

```
((name "Experiment Editor")
(struct (my_toggle          uint8 1)
  (defined          uint8 1)
  (good_guy         uint8 40)
  (bad_guy          uint8 40)
  (label1           uint8 20)
  (maxx             uint8 20)
  (minx             uint8 20)
  (maxy             uint8 20)
  (miny            uint8 20)
  (initial_separation uint8 20)
  (final_separation  uint8 20)
  (seed             uint8 20)
  (DataFile         uint8 20)
)
```

Figure 9. libexp.rdr file – section 1.

```
(editor
  ("Choose the Friendly Vehicle" CHOOSE_ONE good_guy SHOW
    ("HMMWV" "vehicle_US_HMMWV")
    ("M1" "vehicle_US_M1")
  )
  ("Choose the OpFor Vehicle" CHOOSE_ONE bad_guy SHOW
    ("USSR_BMP1" "vehicle_USSR_BMP1")
    ("USSR_BMP2" "vehicle_USSR_BMP2")
    ("USSR_T80" "vehicle_USSR_T80")
  )
  ("Group1" GROUP VERTICAL
    ("Min X (meters)" STRING minx 1)
    ("MAX X (meters)" STRING maxx 1)
    ("DataFile " STRING DataFile 1)
  )
  ("Group2" GROUP VERTICAL
    ("MIN Y (meters)" STRING miny 1)
    ("MAX Y (meters)" STRING maxy 1)
  )
  ("Group3" GROUP VERTICAL
    ("Initial Separation (meters)" STRING initial_separation1)
    ("Final Separation (meters)" STRING final_separation 1)
    ("Random Number Seed" STRING seed 1)
  )
  (" " CHOOSE_ONE my_toggle SHOW
    ("START experiment" 1)
    ("STOP experiment" 2)
  )
)
```

Figure 10. libexp.rdr file – section 2.

```

(initial
  (my_toggle CONSTANT 1)
  (defined CONSTANT 1)
  (bad_guy CONSTANT "vehicle_USSR_BMP1")
  (good_guy CONSTANT "vehicle_US_UGV_S_T_A")
  (label1 CONSTANT "Terrain Parameters")
  (maxx CONSTANT "1000.0" )
  (minx CONSTANT "0.0")
  (maxy CONSTANT "1000.0")
  (miny CONSTANT "0.0")
  (initial_separation CONSTANT "500.0")
  (final_separation CONSTANT "50.0")
  (seed CONSTANT "3713")
  (DataFile CONSTANT "ExpData")

)
(render NOINIT REVERT NEXT)
)

```

Figure 11. libexp.rdr file – section 3.

```

struct exp_edt_struct
{
  uint8 my_toggle;
  uint8 defined;
  char good_guy[40];
  char bad_guy[40];
  char label1[20];
  char maxx[20];
  char minx[20];
  char maxy[20];
  char miny[20];
  char initial_separation[20];
  char final_separation[20];
  char seed[20];
  char DataFile[20];
};

```

Figure 12. The exp_edt_struct structure from the libexp.h file.

byte boundaries. A more detailed discussion of padding and editor structures is in the libeditor documentation (10).

The editor section of the reader file, shown in figure 10, lays out the design of the editor GUI used to control the experiment. This editor has six areas. The first two areas allow the user to choose the vehicles involved in the experiment. The next three areas, labeled Group 1–3, allow the user to choose an area of the battlefield for the experiment and the initial separation of the vehicles. The last section allows users to start and stop the sequence of experiments.

The initial section, shown in figure 11, defines the initial values for each of the variables given in the structure.

Figure 13 shows the GUI for this example editor. Each of the six areas given in figure 11 corresponds to a different column of the editor GUI. Only five of the areas are visible in the figure—the user must use the scroll bar to see the final area. The user can change the layout of the editor by rewriting the editor section of the reader file. It is useful to place frequently changed variables in the upper left portion of the editor to make them easy to find. The fonts, foreground color, and background color used in the editor can be changed using a X-Windows resource file.

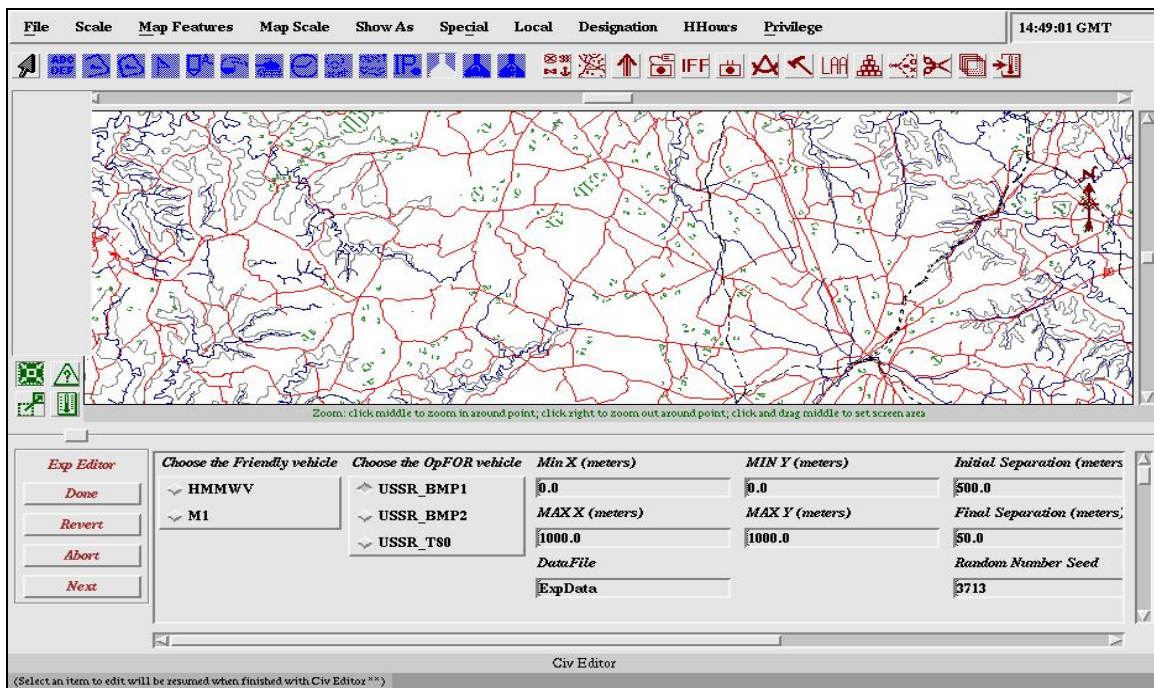


Figure 13. GUI editor for simple meeting engagement experiment.

Although the exact design of the GUI is driven by the specific experimental requirements, there are some features common to all our experimental editors. Each experiment editor has a start and stop button. These allow the user to start a series of experimental trials and to prematurely stop the trials. Every editor includes an experiment location box that allows the user to define the rectangular region of the battlefield to be used for the experimental trials. Since many of the experiments we design involve randomly chosen points and features; the GUI editor allows the user to specify a random number seed. The GUI includes a filename input textbox that is used to construct all the data files for the series. Often editors include boxes to specify unit type, travel distance, and separation distance.

4. Collecting Data

All data will be time-stamped and collected in various files which are specified in the GUI editor as shown in the previous section. While the data collected depends on the specific experimental requirements, a minimal data set for any experiment contains vehicle position and health for each vehicle in the simulation. Also, a minimal data set contains the initial experiment parameters so that we can reconstruct the experiment, if needed.

There are other types of data that are frequently collected in simulation exercises. For example, users may wish to record target acquisitions, along with the available targets (i.e., targets within the vehicle's line of sight [LOS]). Researchers often record vehicle information, such as vehicle state, speed, direction, and subcomponent direction (i.e., sensors). Weapon status, ammunition consumption, and the corresponding damage incurred are also important data items.

Although it is possible to record all the data in a single file, it will be more efficient to record each type of data in separate files. From the editor, users can specify a base file name. The experiment tool uses this name to construct the series of data files recorded for each experiment.

5. Examples

5.1 Simple Meeting Engagement Involving Two Vehicles

In this simple experiment, a single UGV travels through rough terrain, while an enemy vehicle remains stationary. The UGV's mission is to get from point A to point B safely, while the enemy vehicle's mission is to eliminate any enemy vehicles it encounters from its stationary position. The mission continues until the UGV reaches its destination point or sufficient damage has been inflicted. The function *exp_tick* creates the vehicles and runs each trial. The function uses the standard C random number function, *drand48* to vary the UGV's starting location and the enemy vehicle's location.

Setting up this simple scenario involves selecting the vehicle types, placing them on the terrain map, and assigning them their tasks. While this may not seem overly taxing, repeating this process over several iterations can become tedious. The experiment editor allows a researcher to run several replications of a scenario while setting initial parameters only once. This simplifies the setup process for the experiment and also minimizes the amount of user involvement.

5.2 Experiment II Overwatching Fires (OWF) Behavior

The second experiment involves a more complex scenario, also drawn from our robotic behavior development work, and requires a team of manned and armed unmanned systems to protect a

road segment from enemy incursion. The mission, referred to as the OWF, is easy to describe in general terms. There are two distinct roles for team members: observers and shooters. The observers watch for enemy units on the designated road segment. Once enemy units have been identified, the shooters move into position and fire upon enemy units detected by the observers. Once the shooter has fired on its target, it moves to another firing position to await its next target. The mission continues until the enemy unit leaves the road segment; sufficient damage has been inflicted; or the unit receives a new mission.

Thoroughly testing this algorithm requires many experimental trials in which the robots must exercise the behavior algorithms in different situations. The process of setting up and running each experimental trial can be very tedious. A researcher needs to specify a road segment to watch and an observation box for each experimental trial. Not all road segments are suitable—some offer too much or too little nearby cover for a realistic mission scenario. Using an LOS analysis, researchers can find suitable locations for each trial. Once the road segment and observation box are picked, researchers need to set up the friendly and opposing forces and their tasks. Finally the researcher runs the experimental trial. The entire process takes ~10–30 min to find a suitable road segment and set up the experimental trial, and 20 min to run the trial.

We have developed a manager for the OWF experiments; the code is given in appendix D. Figure 14 shows the OWF Experiment Editor, which allows the researcher to set basic parameters for the series of experimental runs. In the top left corner, users enter a base filename, which is used to create the data storage files for the experiments. Users may also specify a random number seed to control repeatability of the experimental trials. There are pull-down menus to select the OFW team and the opposing force. Currently, there are two OFW teams available: 1 “Observer/1 Shooter” and “1 Observer/2 Shooters.” The opposing forces are platoons of BDRMs or BMPs. Finally, the users can restrict the experimental trials to a rectangular region of the battlefield by entering the MaxX, MaxY, MinX, and MinY parameters.

For each trial, the function *find_suitable_road*, given in appendix C, randomly selects road segments and performs an LOS analysis of the nearby terrain. The friendly unit needs open areas to use as firing or observation positions and concealed areas to use as hiding positions and to mask its movement between firing positions. Road segments are rejected if there is not a suitable mix of open and concealed areas near the road. The function *OWF_tick* creates the vehicles and runs each trial. Note that *find_suitable_road* does a preliminary analysis of the terrain to find a suitable region, referred to as the mission box, for an experimental trial. Actual firing and observation positions in the mission box are determined by the overwatching fires behavior.

Trials are terminated after one of the following conditions is satisfied: (1) the opposing force leaves the observed road segment, (2) all of the opposing force is destroyed, or (3) the trial time exceeds the maximum allowed time.

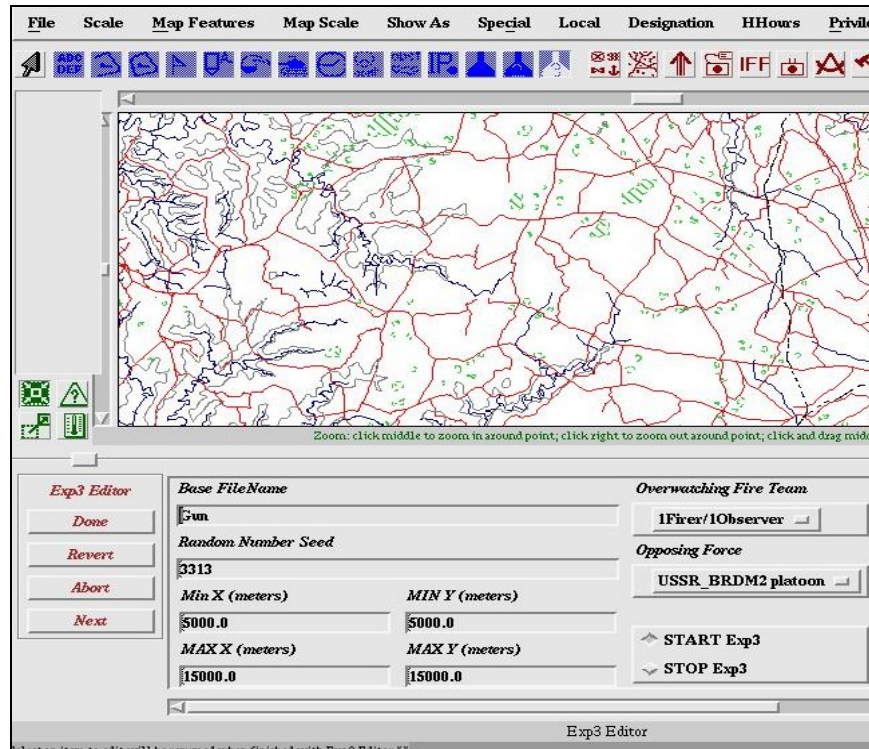


Figure 14. The OWF experiment editor.

The OWF experiment GUI creates a series of data files for later analysis. The first file stores the experimental setup for the trial. Data in this file include the location of the mission box, observation and firing positions, and the observed road segment. Separate files record the positions, health, and activity for the enemy and friendly units during the experimental trial. All the data files begin with the base file name supplied by the user using the OWF experiment GUI. At the end of each trial, the experiment library saves the data files and destroys the current set of vehicles.

The experiment library has been very useful in developing the OWF algorithm. In addition to running complete experiments, it has allowed us to examine our methodology for picking firing and observation positions by generating hundreds of potential positions. In the future, we also want use the same tool to study methods for generating concealed routes between the firing positions.

6. Conclusions

In this report, we have described a method to run multiple simulation trials efficiently using the OneSAF battlefield simulation tool. Our discussion focused on three main areas: (1) managing the experimental trials, (2) specifying experimental parameters with a user interface, and

(3) collecting experimental data. In the previous section, we illustrated the general methodology with two examples.

The experiment editors can save a significant amount of time. For the first example, manual runs take ~3 min (i.e., 1 min to setup the experiment and 2 min for the actual run). Automated runs take ~2.5 min (i.e., 2 min for the actual run and a 30-s delay between the experiments). The real savings is in the amount of time the researcher must spend monitoring the system. In the manual case, the researcher must set up a new run every 3 min. In the automated case, the researcher starts and stops the set of experiments—he may attend to other tasks while the experiments are running.

The time savings is more significant for executing a series of OWF experiments. A single OWF experiment trial takes ~25 min to execute manually (i.e., 10 min to find a suitable road segment and set up the experimental trial and 15 min to run the trial). Automated runs take 16–20 min depending on the complexity of the underlying terrain. Just as in the previous case, automated runs require less supervision so researchers are free to attend to other tasks while the experiments are running.

7. References

1. Smith, J.; Swarts, S. J. *LibSched Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
2. Smith, J. *LibTaskUtil Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
3. Wilbert, D. *LibUnitUtil Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
4. Smith, J.; Courtemanche, A. J. *LibPo Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
5. Smith, J. *LibEntity Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
6. Smith, J. *LibTime Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
7. Smith, J. *LibUnitDefDB Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
8. Braudaway, W.; Buettner, C. G.; Chamberlain, F. L.; Evans, A.; Longtin, M.; Smith, J. E.; Stanzione, T. *LibCTDB Programmer's Reference Manual*; contract no. N00014-92-C-2150; Naval Research Laboratory: Washington, DC, 1996.
9. Smith, J.; Buettner, C. G. *LibCoordinates Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
10. Smith, J. *LibEditor Programmer's Reference Manual*; contract no. N6339-91-D-0001; Naval Air Warfare Center Training Systems Division: Orlando, FL, 1995.
11. Gloeckler F.; Joy, R.; Simpson, J.; Specht, D. *Handbook for Transformation of Datums, Projections, Grids and Common Coordinate Systems*; TEC-SR-7; U.S. Army Topographic Engineering Center: Ft. Belvoir, VA, 1996.

INTENTIONALLY LEFT BLANK.

Appendix A. The UUTIL_UNIT_PARAMS Structure

This appendix appears in its original form, without editorial change.

```

typedef struct util_unit_params
{
    char                *formation;                                /* line, vee, wedge*/
    char                *sub_formation;                            /* line, vee, wedge*/
    ForceID             force;                                         /* Friendly, Opposing, Neutral */
    uint8               methodology;                                   /* OneSAF, Soar */
    uint8               mounted;                                       /* True = Mounted, False = Dismounted */
    uint8               _pad1;
    ObjectType         object_type;
    VehicleMarking      marking;                                       /* Radio call sign */
    TopUnitMarking      top_unit;                                     /* Radio call sign for supervisor*/
    uint8               _pad2;
    EchelonRolerole;
    uint16 _pad3;
    float64            location[XYZC];                               /* used in unit editor as location */
    Angle              direction;                                  /* with respect to the location */
    float32             competence;                                    /* driving ability */
    UTIL_EQUIP_DATA     equip[UUTIL_MAX_VEHICLE_TYPES];
    uint16              use_default_munitions;
    uint16              _pad4;
    SimulationAddress   commander;
    int32               damage;
    int32               repair;
    AmpCrftInventory    craft[maxAmphibCraft];
    /* The following parameters are for creating groups of pallets only: */
    int32               number_to_create;
    float32             pallet_row_length;
    float32             pallet_row_distance;
    uint8               pallet_row_type; /* single or double */
    uint8               _padding1;
    uint16              _padding2;
    ObjectID            superior;
    uint16              _pad5;
    int32               posture;
    int32               weapon;
    int32               action;
    uint32              _pad6;
    float64             sensor_location[XYZC];                       /* Used in unit editor*/
} UTIL_UNIT_PARAMS;

```

Appendix B. The M1 Experiment Editor

This appendix appears in its original form, without editorial change.

```

static void M1_Experiment (PO_DATABASE *db)
{
    /* Variable definition */
    PO_DB_ENTRY *frame, *point_entry, M1Unit _entry;
    int32 vehicle_id, methodology, sx, sy, ex, ey;
    float64 speed, angle;
    uint32 size;
    char      *otype, unit_type [80], formation [80];
    ObjectType object;
    ObjectID   M1Unit_ID;
    UnitClass  M1Unit;
    UUTIL_UNIT_PARAMS new_M1Unit;
    PO_BUFFER (PointClass, point, pointbuf);

    /*1. Find the object from the unit name and the unit_db*/
    otype = stdname_get_standard (reader_get_symbol ("unit_US_M1_Platoon")); object =
        unitdefdb_name_to_object_type (otype, unit_db);

    /*2. Set up the unit parameters structure */
    bzero (&new_M1Unit, sizeof (UUTIL_UNIT_PARAMS));
    M1Unit.force = distinguishedForceID;
    M1Unit.methodology = 0;
    M1Unit.mounted = FALSE;
    strcpy ((char *) formation, "line");
    M1Unit.formation = reader_get_symbol (formation);
    M1Unit.object_type = object;
    sx = 5000.0 + 1000.0*drand48 ();    sy = 5000.0 + 1000.0*drand48 ();
    ex = sx + 500.0*drand48();          ey = sy + 500.0*drand48 ();
    M1Unit.location[0] = sx;            M1Unit.location[1] = sy;
    angle = atan((eby - sby)/(ebx - sbx));
    M1Unit.direction = RAD_TO_BAM(angle);
    M1Unit.competence = 1.0;
    M1Unit.use_default_munitions = TRUE;
    M1Unit.sub_formation = reader_get_symbol("line");

    /*3. Create the M1 platoon */
    M1Unit _entry = unitutil_create(&M1Unit, NULL,0,NULL, 0, TRUE, db);
    M1Unit _unit = PO_UNIT_DATA(M1Unit _entry);
    overlay = po_get_object(db, &PO_UNIT_DATA(M1Unit _entry).overlayID);

    /* 4. Create a point for the move mission */
    bzero(&point, sizeof(point));
    point.overlayID = PO_UNIT_DATA(M1Unit _entry).overlayID;
    point.style = PSgeneral;
    point.color = OCGreen;
    point.dashed = 0;
    point.location.x = ex;    point.location.y = ey;
    point.direction = RAD_TO_BAM(DEG_TO_RAD((float64) degrees));
    point_entry = po_create_object(db, NULL, objectClassPoint,
        FALSE, &point, sizeof(point), NULL);

    /* 5. Create the Move Mission */
    frame = taskutil_create_frame("Move");
    taskutil_set_parameter(frame,SM_UMixedTravel,"route",
        &PO_OBJECT_ID(point_entry));

    speed = 20.0;
    taskutil_set_parameter(frame,SM_UMixedTravel,"speed", &speed);
    M1Unit_ID = PO_OBJECT_ID(M1Unit _entry);
    taskutil_assign(frame,& M1Unit_ID);
}

```

Appendix C. The Reader File for Editor Shown in Figure 9

This appendix appears in its original form, without editorial change.

```

;; Editor definition for the Experiment editor
;; Section One
(
  (name "Experiment Editor")
  (struct (my_toggle uint8 1)
    (defined uint8 1)
    (good_guy uint8 40)
    (bad_guy uint8 40)
    (label1 uint8 20)
    (maxx uint8 20)
    (minx uint8 20)
    (maxy uint8 20)
    (miny uint8 20)
    (initial_separation uint8 20)
    (final_separation uint8 20)
  )
)
;; Section Two
(editor
  ("Choose the Good Guy" CHOOSE_ONE good_guy SHOW
    ("HMMWV" "vehicle_US_HMMWV")
    ("M1" "vehicle_US_M1"))
  ("Choose the Bad Guy" CHOOSE_ONE bad_guy SHOW
    ("USSR_BMP1" "vehicle_USSR_BMP1")
    ("USSR_BMP2" "vehicle_USSR_BMP2")
    ("USSR_T80" "vehicle_USSR_T80"))
  ("Group1" GROUP VERTICAL
    ("Min X (meters)" STRING minx 1)
    ("MAX X (meters)" STRING maxx 1))
  ("Group2" GROUP VERTICAL
    ("MIN Y (meters)" STRING miny 1)
    ("MAX Y (meters)" STRING maxy 1))
  ("Group3" GROUP VERTICAL
    ("Initial Separation (meters)" STRING initial_separation 1)
    ("Final Separation (meters)" STRING final_separation 1))
  (" " CHOOSE_ONE my_toggle SHOW
    ("START experiment" 1)
    ("STOP experiment" 2))
)
;; Section Three
(initial
  (my_toggle CONSTANT 1)
  (defined CONSTANT 1)
  (bad_guy CONSTANT "vehicle_USSR_BMP1")
  (good_guy CONSTANT "vehicle_US_UGV_S_T_A")
  (label1 CONSTANT "Terrain Parameters")
  (maxx CONSTANT "1000.0" )
  (minx CONSTANT "0.0")
  (maxy CONSTANT "1000.0")
  (miny CONSTANT "0.0")
  (initial_separation CONSTANT "500.0")
  (final_separation CONSTANT "50.0")
)
(render NOINIT REVERT NEXT)
)

```

Appendix D. The Overwatching Fires Experiment Editor

This appendix appears in its original form, without editorial change.

```

#ifndef lint
static char rcsid [ ] = "$RCSfile$ $Revision$ $State$";
#endif
/*****
* File: exp3_init.c
*****/

#include "libexp3_local.h"
#include <Xm/Xm.h>
#include <Xm/PushButton.h>
#include <Xm/RowColumn.h>
#include <Xm/Frame.h>
#include <Xm/Label.h>
#include <Xm/Text.h>
#include <Xm/ToggleB.h>
#include <Xm/ScrollBar.h>
#include <p_size.h>
#include <po_size.h>
#include <libpo.h>
#include <stdalloc.h>
#include <stdext.h>
#include <stdlib.h>
#include <stddef.h>

extern time_t time();

#include "exp3_icon.h"

#define BARNAME "Exp3"
#define DEFINED_TO_NOW 2

static UNITDEFDB_DB unit_db;
EDT_EDITOR_PTR result;
/* Borrowed from other sections of OneSAF */
struct overlay_search
{
    int32found;
    ObjectID*id;
    PO_DB_ENTRY *entry;
};

static void get_an_overlay(PO_DATABASE *db,ObjectID*id);
static void find_an_overlay(PO_DB_ENTRY *entry, struct overlay_search *search);
static void expand_route_in_place (CTDB *ctdb, ROUTE_LIST *rlp);

float64 Mymax_x, Mymin_x, Mymax_y, Mymin_y;
long int Myseed;
int32 tickNumber;
char dir_base_name[10], dir_name[20], elev[60], los[60], GenFileName[60], cmd;
FILE *GenFile,*elevFile,*losFile, *goodPos, *badPos;
EXP3_EDITOR *edtr;

void draw_box( PO_DATABASE *db, int32 color,int32 number,
    float64 x0,float64 y0, float64 x1,float64 y1,
    float64 x2,float64 y2, float64 x3,float64 y3);

void make_box( float64 Cx, float64 Cy,float64 distX, float64 distY, float64 dX, float64 dY,

```

```

float64 *x0,float64 *y0, float64 *x1,float64 *y1,
float64 *x2,float64 *y2, float64 *x3,float64 *y3);

static PO_DB_ENTRY *make_line( PO_DATABASE *db, int numPoints,
float64 *x,float64 *y, int32 color);
static PO_DB_ENTRY *make_point( PO_DATABASE *db,
ObjectID *overlay, float64 x, float64 y, float64 direction);
static int32 exp3_alloc(EXP3_EDITOR*edtr, char *name, ObjectID *id);
static void exp3_dealloc(EXP3_EDITOR*edtr, int32 which);
static void exp3_done(EDT_EDITOR_PTReditor, struct exp3_edt_struct *new,
EXP3_EDITOR*edtr, EDT_EXIT_STATUS status);
static void exp3_render(EDT_EDITOR_PTReditor, int32 transient,
struct exp3_edt_struct *old, struct exp3_edt_struct *new, EXP3_EDITOR*edtr);
static void new_or_changed_handler(PO_DB_ENTRY*entry, EXP3_EDITOR*edtr);
static void gone_handler(PO_DB_ENTRY*entry, EXP3_EDITOR*edtr);
static void exp3_selected(Widgetw, EXP3_EDITOR*edtr, XmPushButtonCallbackStruct *call_data);
static void handle_local_force_event(int32 promoted_force, int32 display, EXP3_EDITOR*edtr);
static int32 exp3_find_suitable_road(PO_DATABASE *db,
float64 XMin, float64 YMin, float64 XMax, float64 YMax,
float64 *tx0,float64 *ty0, float64 *tx1,float64 *ty1,
float64 *tx2,float64 *ty2, float64 *tx3,float64 *ty3,
float64 *Obsx0,float64 *Obsy0, float64 *Obsx1,float64 *Obsy1,float64 *Obsx2,float64 *Obsy2, float64
*Obsx3,float64 *Obsy3,
float64 *gMinX,float64 *gMinY, float64 *gMaxX,float64 *gMaxY,
float64 *StartX,float64 *StartY, float64 *StopX,float64 *StopY);

void exp3_march_along_road(PO_DATABASE *db,float64 sx,float64 sy,float64 step,
int32 max_points, float64 initialDx,float64 initialDy,
float64 *CurrentRoadX,float64 *CurrentRoadY);
int32 exp3_time_to_quit(PO_DATABASE *db);
void dump_terrain_surface( char *file_name,
float64 MinX,float64 MinY, float64 MaxX,float64 MaxY);
static exp3_count = 0;
float64 sgx,sgy,sbx,sby,zz,vec[2], px,py,rx,ry,egx,egy,ebx,eby,angle1, speed, degrees;
float64 tx0,ty0,tx1,ty1,tx2,ty2,tx3,ty3, tlength,twidth,tdx,tdy;
float64 Obsx0,Obsy0,Obsx1,Obsy1,Obsx2,Obsy2,Obsx3,Obsy3;
float64 Duration, my_roadX[150],my_roadY[150], observedRoadX[50],observedRoadY[50];
float64 entranceRoadX[50],entranceRoadY[50], exitRoadX[10],exitRoadY[10];
int32 found_start_point,found_end_point;
CTDB *ctdb;

UTIL_UNIT_PARAMS new_good_guy, new_bad_guy;
PO_DB_ENTRY *BadGuyRouteEntry;
PO_DB_ENTRY *good_guy_entry,*bad_guy_entry,*overlay,*overlay_g,*point_entry,
*point1_entry,*point2_entry;

static void exp3_tick_good(PO_DATABASE *db)
{
VehicleID simnet_id;
int32vehicle_id;
static float64 random_number;
RGMOVE_PARAMETERS *my_utavel_parameters,*g_utavel_parameters;
RGUNNERY_PARAMETERS *my_rgunnery_parameters;
char my_task_name[]={"OWFexp"},exp3_label[100];
int32 methodology, x, y, degrees;
uint32 size;

```

```

char task_type[80],unit_type[80], marking[80], formation[80];
ObjectType country_type;
char*otype;
ObjectType object;
ObjectID good_guy_id,*good_guy_overlay_id;
ObjectType my_unit_type;
UnitClass good_guy_unit;
PointClass point,point1,point2;
float64 Obox_length,Obox_width,Obox_direction,Tbox_length,Tbox_width,Tbox_direction;
char value[80];
int32 UsingEditor = 0;
PO_DB_ENTRY *Gframe;
int32 i,current_time,location_number, test,is_prep, is_act;
uint32 primary_prep_model, primary_act_model, model;
char MyLosFile[20], MissionDuration[20];

Obox_length = sqrt((Obsx0 - Obsx3)*(Obsx0 - Obsx3) + (Obsy0 - Obsy3)*(Obsy0 - Obsy3));
Obox_width = sqrt((Obsx0 - Obsx1)*(Obsx0 - Obsx1) + (Obsy0 - Obsy1)*(Obsy0 - Obsy1));
Obox_direction = atan( (Obsy0 - Obsy1)/((Obsx0 - Obsx1) + 0.001));
Obox_direction = 45.0/atan(1.0)*Obox_direction;
Tbox_length = sqrt((tx0 - tx3)*(tx0 - tx3) + (ty0 - ty3)*(ty0 - ty3));
Tbox_width = sqrt((tx0 - tx1)*(tx0 - tx1) + (ty0 - ty1)*(ty0 - ty1));
Tbox_direction = atan( (ty0 - ty1)/((tx0 - tx1) + 0.001));
Tbox_direction = 45.0/atan(1.0)*Tbox_direction;
Duration = 5.0;

ctdb = (CTDB *)gcs_get_tdb(GCS_ILLEGAL_CELL, FALSE);

/* the good_guy */

strcpy((char *)unit_type, good_guy);
sprintf(unit_type,"unit_US_OWF_Team_1",good_guy);
sprintf(exp3_label,"G%d",exp3_count);
strcpy((char *)marking, exp3_label);
methodology = 0;
strcpy((char *)formation, "wedge");

otype = stdname_get_standard(reader_get_symbol("unit_US_OWF_Team_1"));
object = unitdefdb_name_to_object_type(otype,unit_db);

bzero(&new_good_guy, sizeof(UUTIL_UNIT_PARAMS));
new_good_guy.force = distinguishedForceID;
new_good_guy.methodology = methodology;
new_good_guy.mounted = FALSE;
new_good_guy.formation = reader_get_symbol(formation);
new_good_guy.object_type = object;
strcpy((char *)new_good_guy.marking.text, (char *)marking);
new_good_guy.location[0] = (Obsx1 + Obsx2)/2.0;
new_good_guy.location[1] = (Obsy1 + Obsy2)/2.0;
new_good_guy.direction = RAD_TO_BAM(Obox_direction);
new_good_guy.competence = .5;
new_good_guy.use_default_munitions = TRUE;
new_good_guy.sub_formation = reader_get_symbol("line");

good_guy_entry = unitutil_create(&new_good_guy, NULL,0,NULL,0, TRUE, db);
good_guy_unit = PO_UNIT_DATA(good_guy_entry);

```

```

    roe_set_permission(good_guy_entry, freePermission, ROE_propagate_fields);
    overlay_g = po_get_object(db, &PO_UNIT_DATA(good_guy_entry).overlayID);
    good_guy_overlay_id = &PO_UNIT_DATA(good_guy_entry).overlayID;

    bzero(&point1, sizeof(point1));
    point1.overlayID = PO_UNIT_DATA(good_guy_entry).overlayID;
    point1.style = PStarget;
    point1.color = OCYellow;
    point1.dashed = 0;
    point1.location.x = (tx0 + tx1 + tx2 + tx3)/4.0;
    point1.location.y = (ty0 + ty1 + ty2 + ty3)/4.0;
    point1.direction = RAD_TO_BAM(angle1);
    point1_entry = po_create_object(db, NULL, objectClassPoint,
        FALSE, &point1, sizeof(point1), NULL);

    /* Create the actual frame */
    Gframe = taskutil_create_frame("RGunnery");
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targx0", &tx0);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targy0", &ty0);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targx1", &tx1);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targy1", &ty1);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targx2", &tx2);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targy2", &ty2);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targx3", &tx3);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Targy3", &ty3);

    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsx0", &Obsx0);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsy0", &Obsy0);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsx1", &Obsx1);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsy1", &Obsy1);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsx2", &Obsx2);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsy2", &Obsy2);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsx3", &Obsx3);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Obsy3", &Obsy3);
    taskutil_set_parameter(Gframe, SM_RGunnery, "Duration", &Duration);
    taskutil_set_parameter(Gframe, SM_RGunnery, "UsingEditor", &UsingEditor);
    taskutil_set_parameter(Gframe, SM_RGunnery, "enemyCG", &PO_OBJECT_ID(point1_entry));
    taskutil_set_parameter(Gframe, SM_RGunnery, "ExerciseNumber", &exp3_count);
    good_guy_id = PO_OBJECT_ID(good_guy_entry);
    taskutil_assign(Gframe, &good_guy_id);
    tickNumber = 1;
}

static void exp3_tick_bad(PO_DATABASE *db)
{
    VehicleID simnet_id;
    PO_DB_ENTRY *frame;
    int32 vehicle_id;
    float64 speed;
    static float64 random_number;
    float64 pi, angle, radius;
    UMXTRAVEL_PARAMETERS *my_utrael_parameters, *g_utrael_parameters;
    char my_task_name[] = {"BAD TASK"};
    char exp3_label[100];
    int32 user_spacing, methodology, x, y;

```

```

float64      y_offset,follow_dist;
uint32 size;
char task_type[80],unit_type[80], marking[80], formation[80];
ObjectType country_type;

char*otype;
ObjectType object;
ObjectID bad_guy_id,bad_guy_overlay_id;
ObjectType my_unit_type;
UnitClassbad_guy_unit;
PointClass point,point1,point2;
PO_DB_ENTRY *line_entry, *box_entry,*line1_entry;
UNITDEFDB_UNIT db_bad_guy_unit;
const char*bad_guy_name;

/* Bad guy */
PO_BUFFER(LineClass, line, linebuf);
PO_BUFFER(LineClass, box, boxbuf);
int32 i,current_time,location_number;
int32 test,is_prep, is_act;
uint32 primary_prep_model, primary_act_model, model;
PO_DB_ENTRY *vehicle_entry[10];
int32 n_member,useRoads = True;
VehicleID followerID= {{0, 0}, 0};

pi = 4.0*atan(1.0);
test = 0;

speed = 20.0;

current_time = time_simulation_time();

/* BAD GUY STUFF */
/* refer to libsafmark */

strcpy((char *)unit_type, bad_guy);
sprintf(exp3_label,"B%d",exp3_count);
strcpy((char *)marking, exp3_label);
methodology = 0;
strcpy((char *)formation, "line");

otype = stdname_get_standard(reader_get_symbol("unit_USSR_BMP2_Platoon"));
object = unitdefdb_name_to_object_type(otype,unit_db);

bzero(&new_bad_guy, sizeof(UUTIL_UNIT_PARAMS));

new_bad_guy.force = otherForceID;
new_bad_guy.methodology = methodology;
new_bad_guy.mounted = TRUE;
new_bad_guy.formation = reader_get_symbol(formation);
new_bad_guy.object_type = object;
strcpy((char *)new_bad_guy.marking.text, (char *)marking);
new_bad_guy.location[0] = my_roadX[0];
new_bad_guy.location[1] = my_roadY[0];
angle = atan((my_roadY[1] - my_roadY[0])/(my_roadX[1] - my_roadX[0]));
new_bad_guy.direction = RAD_TO_BAM(angle);

```

```

new_bad_guy.competence = 1.0;
new_bad_guy.use_default_munitions = TRUE;
new_bad_guy.sub_formation = reader_get_symbol("line");

bad_guy_entry = unitutil_create(&new_bad_guy, NULL,0,NULL, 0, TRUE, db);
bad_guy_unit = PO_UNIT_DATA(bad_guy_entry);

roe_set_permission(bad_guy_entry, holdPermission,ROE_propagate_fields);

overlay = po_get_object(db, &PO_UNIT_DATA(bad_guy_entry).overlayID);

/* Create a point for the overlay */

printf("EXP3 done creating point for bad guy ..... \n");

/* Create the actual frame */
frame = taskutil_create_frame("Move");
taskutil_set_parameter(frame,SM_UMixedTravel,"route",
&PO_OBJECT_ID(BadGuyRouteEntry));

speed = 13.0;
y_offset = 5.0;

taskutil_set_parameter(frame,SM_UMixedTravel,"speed", &speed);
taskutil_set_parameter(frame,SM_UMixedTravel,"roadmarch", &useRoads);
taskutil_set_parameter(frame,SM_UMixedTravel,"formation", &formation);
taskutil_set_parameter(frame,SM_UMixedTravel,"follow_dist", &y_offset);
taskutil_set_parameter(frame,SM_UMixedTravel,"leader", &followerID);

bad_guy_id = PO_OBJECT_ID(bad_guy_entry);
taskutil_assign(frame,&bad_guy_id);

t1 = time_last_realtime_clock;
my_time = (float) t1/1000.0;
}

static void exp3_tick(PO_DATABASE *db)
{
    float64 pi,angle,radius,Gx,Gy;
    float64 gMinX,gMinY,gMaxX,gMaxY;
    float64 dist_good, dist_bad, time_good, time_bad;
    int32 my_millisecond;
    int32 grade;
    int32 end_seg, esi;
    char goodFileName[20],badFileName[20];
    CTDB *out_ctdb;

    ctdb = (CTDB *)gcs_get_tdb(GCS_ILLEGAL_CELL, FALSE);
    speed = 20.0;
    exp3_count = exp3_count + 1;

    /* 1. find a road segment for the enemy units to use */
    grade = exp3_find_suitable_road( db,Mymin_x,Mymin_y,Mymax_x,Mymax_y,

```

```

        &tx0,&ty0, &tx1,&ty1,&tx2,&ty2, &tx3,&ty3,&Obsx0,&Obsy0,
        &Obsx1,&Obsy1,&Obsx2,&Obsy2, &Obsx3,&Obsy3,
        &gMinX,&gMinY,&gMaxX,&gMaxY,&sbx,&sby,&ebx,&eby);

    sprintf(GenFileName,"%sGen.%d",dir_base_name,exp3_count);
    sprintf(goodFileName,"%sGood.%d",dir_base_name,exp3_count);
    sprintf(badFileName,"%sBad.%d",dir_base_name,exp3_count);
    printf("My file is %s",GenFileName);
    GenFile = fopen(GenFileName,"w");
    goodPos = fopen(goodFileName,"w");
    badPos = fopen(badFileName,"w");
    fprintf(GenFile, "Target (%.3lf, %.3lf) (%.3lf, %.3lf) (%.3lf, %.3lf) \n",
        tx0,ty0,tx1,ty1,tx2,tx2,tx3,tx3);
    fprintf(GenFile, "Obs (%.3lf, %.3lf) (%.3lf, %.3lf) (%.3lf, %.3lf) \n",
        Obsx0,Obsy0,Obsx1,Obsy1,Obsx2,Obsx2,Obsx3,Obsx3);
    fprintf(GenFile, "BoundingBox (%.3lf, %.3lf) (%.3lf, %.3lf)\n",
        gMinX,gMinX,gMaxX,gMaxX);
    fclose(GenFile);
    dump_terrain_surface(dir_base_name, gMinX,gMinY,gMaxX,gMaxY);

    tdx = tx3 - tx0;
    tdy = ty3 - ty0;
    tlength = sqrt(tdx*tdx + tdy*tdy);
    tdx = tdx/(tlength + 0.001); tdy = tdy/(tlength + 0.001);

    twidth = sqrt((tx1 - tx0)*(tx1 - tx0) + (ty1 - ty0)*(ty1 - ty0));

    Gx = (tx0 + tx1 + tx2 + tx3 + Obsx0 + Obsx1 + Obsx2 + Obsx3)/8.0;
    Gy = (ty0 + ty1 + ty2 + ty3 + Obsy0 + Obsy1 + Obsy2 + Obsy3)/8.0;
    tactmap_set_center(edtr->tactmap, (int32) tx0, (int32) ty0, GCS_ILLEGAL_CELL);
    draw_box(db,OCBlack,exp3_count,tx0,ty0,tx1,ty1,tx2,ty2,tx3,ty3);
    draw_box(db,OCBlack,exp3_count,Obsx0,Obsy0,Obsx1,Obsy1,Obsx2,Obsy2,Obsx3,Obsy3);

    /* schedules the good guy and bad guy events */
    printf("scheduling bad guy \n");
    sched_deferred_fncl((SCHED_FUNCTION)exp3_tick_bad,10, my_groupG,A_PTR,db,A_END);
    sched_deferred_fncl((SCHED_FUNCTION)exp3_tick_good,10000, my_groupB,A_PTR,db,A_END);

}
/* MAF */

/*ARGSUSED*/
static void exp3_edit(
    SGUI_PTRgui,
    SGUI_MODE_PTR mode,
    EDT_EDITOR_PTReditor,
    SGUI_MODE_STATE state)
{
    edt_state(editor, mode, state);
}

void exp3_init(UNITDEFDB_DB unitdefdb)
{
    unit_db = unitdefdb;
}

```

```

int32 exp3_init_gui(
    char*data_path,
    uint32 reader_flags,
    SGUI_PTR gui,
    TACTMAP_PTR tactmap,
    COORD_TCC_PTR tcc,
    GC map_erase_gc,
    SNSTVE_WINDOW_PTR sensitive,
    CALLBACK_EVENT_PTR refresh_event,
    PO_DATABASE *db)
{
    READER_UNION data;
    ObjectID dummy;
    UnitClassunit;
    int32 exp3;
    int32 which;

    int32 ret = reader_read("exp3.rdr", data_path, &data,
        reader_flags & ~READER_TYPING);

    ctdb = (CTDB *)gcs_get_tdb(GCS_ILLEGAL_CELL, FALSE);
    ctdb_get_extent( ctdb,&gmin_x,&gmin_y,&gmax_x,&gmax_y);

    if (ret)
        return ret;

    edtr = (EXP3_EDITOR *)STDALLOC(sizeof(EXP3_EDITOR));
    bzero((char *)edtr, sizeof(EXP3_EDITOR));

    edtr->gui = gui;
    edtr->tactmap = tactmap;
    edtr->map_erase_gc = map_erase_gc;
    edtr->sensitive = sensitive;
    edtr->db = db;

    edtr->editor = edt_create(data.array,
        sizeof(struct exp3_edt_struct),
        /* callback */
        (EDT_RENDER_FUNCTION) exp3_render, (ADDRESS) edtr,
        (EDT_RENDER_FUNCTION) exp3_render, (ADDRESS) edtr,
        /* callback */
        (EDT_EXIT_FUNCTION) exp3_done, edtr,
        TRUE, gui, tactmap, map_erase_gc, sensitive, refresh_event, db);
    edtr->mode =
        sgui_add_mode(gui, SGUI_OBJECT_MODE,
            (char *) exp3_icon_bits, exp3_icon_width, exp3_icon_height,
            /* callback */
            (SGUI_MODE_CALLBACK)exp3_edit, edtr->editor,
            "Exp3 Editor", "Exp3 Editor **");

    /* Make our first exp3 button now, to reserve our place on the
    * menubar.
    */

```

```

/* Post callbacks for exp3 objects */
callback_register_handler(db->new_object_event,
    (CALLBACK_HANDLER) new_or_changed_handler, edtr);
callback_register_handler(db->object_changed_event,
    (CALLBACK_HANDLER) new_or_changed_handler, edtr);
callback_register_handler(db->object_gone_event,(CALLBACK_HANDLER) gone_handler, edtr);

/* Register a handler for privilege local force changing */
callback_register_handler(
    priv_get_local_force_event(sgui_get_priv(edtr->gui)),
    (CALLBACK_HANDLER) handle_local_force_event, edtr);

return 0;
}

/* exp3_done (a callback routine) gets called whenever the user presses
 * 'Done' or 'Abort' from the exp3 editor.
 */
/*ARGSUSED*/
static void exp3_done(
    EDT_EDITOR_PTR editor,
    struct exp3_edt_struct *new,
    EXP3_EDITOR*edtr,
    EDT_EXIT_STATUS status)
{
    int i;
    PO_DB_ENTRY *entry;
    SCHED_FNCL_PTR my_funct;

    /* Create the object if the Done button was pressed, otherwise do nothing*/
    my_group = 9976;
    my_group = 1173;
    my_group = 4621;
    if (status == EDT_DONE)
    {
        toggle = new->my_toggle;
        sscanf(new->seed,"%ld",&Myseed);
        printf("here with seed = %ld\n",Myseed);
        sscanf(new->minx,"%lf",&Mymin_x);
        sscanf(new->maxx,"%lf",&Mymax_x);
        sscanf(new->miny,"%lf",&Mymin_y);
        sscanf(new->maxy,"%lf",&Mymax_y);
        printf("(%.3f, %.3f) - (%.3f, %.3f) \n",Mymin_x,Mymin_y,Mymax_x,Mymax_y);
        sscanf(new->Trials,"%d",&Trials);
        sscanf(new->TrialDuration,"%lf",&TrialDuration);
        sscanf(new->good_guy,"%s",good_guy);
        sscanf(new->bad_guy,"%s",bad_guy);
        srand48(Myseed);
        printf("coming in with %s ",new->base_file_name);
        sprintf(dir_base_name,"%s",new->base_file_name);
        printf("leaving with %s ",dir_base_name);
        if (toggle == 1)
        {
            tickNumber = -99999999;
            sched_deferred_fncl((SCHED_FUNCTION) exp3_schedule_tick,90, my_group,
                A_PTR,edtr->db,A_END);
        }
    }
}

```

```

        sched_deferred_fncl((SCHED_FUNCTION) exp3_tick,90, my_group,
                           A_PTR,edtr->db,A_END);
    }
}
else if (status == EDT_ABORT)
{
    printf("EXP3 rying to cancel the tick \n");
}
}

static void exp3_render(
    EDT_EDITOR_PTReditor,
    int32 transient,
    struct exp3_edt_struct *old,
    struct exp3_edt_struct *new,
    EXP3_EDITOR*edtr)
{
    if (new->defined == 0)
    {
        struct exp3_edt_struct copy;
        sprintf(copy.base_file_name,"Gun");
        sprintf(copy.minx,"%0.1lf",Mymax_x);
        sprintf(copy.maxx,"%0.1lf",Mymax_x);
        sprintf(copy.miny,"%0.1lf",Mymax_y);
        sprintf(copy.maxy,"%0.1lf",Mymax_y);
        sprintf(copy.seed,"%ld",&Myseed);
        copy.defined = 1;
        strcpy(copy.good_guy,new->good_guy);
        strcpy(copy.bad_guy,new->bad_guy);
        strcpy(copy.Trials, new->Trials);
        strcpy(copy.TrialDuration, new->TrialDuration);
        printf("in render \n");
        printf("coming in with %s ",new->base_file_name);
        sprintf(dir_base_name,"%s",new->base_file_name);
        edt_load(edtr->editor, &copy, sizeof(copy));
    }
}

/* Allocate a button on the menubar.
*/
static int32 exp3_alloc(
    EXP3_EDITOR*edtr,
    char *name,
    ObjectID *id)
{
    int32 this = edtr->n_menu;
    int32 i;
    XmString xms;
    Arg args[1];

    /* Make sure we don't post the same thing twice */
    for (i=0;i<edtr->n_menu;i++)
    if (NS_OBJECT_IDS_EQUAL(edtr->menu[i].hhour_id, *id))
    {
        this = i;
        break;
    }
}

```

```

    }

    if (this == edtr->n_menu)
    {
        edtr->n_menu += 1;
        if (!edtr->menu)
        {
            edtr->menu = (struct exp3_menu *)
                STDALLOC(sizeof(struct exp3_menu));
            edtr->menu_size = 1;
        }
        else if (edtr->n_menu > edtr->menu_size)
        {
            edtr->menu_size = edtr->n_menu;
            edtr->menu = (struct exp3_menu *)
                STDREALLOC(edtr->menu,
                    sizeof(struct exp3_menu) * edtr->menu_size);
        }
        edtr->menu[this].button =
            sgui_add_menubar(edtr->gui, PRIV_COMMANDER, BARNAME,
                "Traffic", xmPushButtonWidgetClass);
        edtr->menu[this].hhour_id = *id;

        /* Post a handler */
        XtAddCallback(edtr->menu[this].button, XmNactivateCallback,
            (XtCallbackProc) exp3_selected, edtr);
    }

    xms = XmStringCreateLtoR(name, XmSTRING_DEFAULT_CHARSET);
    XtSetArg(args[0], XmNlabelString, xms);
    XtSetValues(edtr->menu[this].button, args, 1);
    XmStringFree(xms);

    return this;
}

/* Remove a button from the menubar (note that we do not reuse buttons
 * because the user probably expects these to be in the order received).
 */
static void exp3_dealloc(
    EXP3_EDITOR*edtr,
    int32 which)
{
    int32 i;

    XtUnmanageChild(edtr->menu[which].button);
    XtDestroyWidget(edtr->menu[which].button);

    for (i=which; i<edtr->n_menu-1; i++)
        edtr->menu[i] = edtr->menu[i+1];
    edtr->n_menu--;
}

/* Handle new exp3
 */

```

```

static void new_or_changed_handler(
    PO_DB_ENTRY*entry,
    EXP3_EDITOR*edtr)
{
    PO_DB_ENTRY *unit;
    int32 which;

    /* Ignore stuff we don't want */
    if ((PO_OBJECT_CLASS(entry) != objectClassHHour) || (PO_HHOUR_DATA(entry).forceID ==
    forceIDIrrelevant))
        return;

    /* Make a button (if there already is a button, this will find it) */
    which = exp3_alloc(edtr, PO_HHOUR_DATA(entry).name, &PO_OBJECT_ID(entry));

    /* Make sure the button is unmanaged if it doesn't match the
    * local force setting, and managed if it does.
    */
    if (!priv_get_local_force(sgui_get_priv(edtr->gui),
        PO_HHOUR_DATA(entry).forceID))
    {
        XtUnmanageChild(edtr->menu[which].button);
    }
    else
    {
        XtManageChild(edtr->menu[which].button);
    }
}

/* Handle exp3 going away.
*/
static void gone_handler(
    PO_DB_ENTRY*entry,
    EXP3_EDITOR*edtr)
{
    int32 i;

    /* Ignore stuff we don't want */
    if ((PO_OBJECT_CLASS(entry) != objectClassHHour) ||
        (PO_HHOUR_DATA(entry).forceID == forceIDIrrelevant))
        return;

    /* Is this in our list? */
    for (i=0;i<edtr->n_menu;i++)
        if (NS_OBJECT_IDS_EQUAL(edtr->menu[i].hhour_id, PO_OBJECT_ID(entry)))
        {
            exp3_dealloc(edtr, i);
            return;
        }
}

/* Handle exp3 being selected from the menu bar */
static void exp3_selected(
    Widgetw,
    EXP3_EDITOR*edtr,
    XmPushButtonCallbackStruct *call_data)
{

```

```

int32 i;
PO_DB_ENTRY *entry;
HHourClass *po_data;
struct exp3_edt_struct copy;

for (i=0;i<edtr->n_menu;i++)
    if (w == edtr->menu[i].button)
        break;

if (i==edtr->n_menu)
    return;

if (NS_OBJECT_ID_NULL(edtr->menu[i].hhour_id) ||
    !(entry = po_get_object(edtr->db, &edtr->menu[i].hhour_id)) ||
    (PO_OBJECT_CLASS(entry) != objectClassHHour))
    return;

sgui_set_mode(edtr->mode);

/* Remember what we're editing */
/* Load it with initial values */
edt_load(edtr->editor, &copy, sizeof(copy));
}

/* Handle which exp3 should be available to be selected from the menu bar */
static void manage_exp3_force(EXP3_EDITOR *edtr)
{
    int32 iloop;
    PO_DB_ENTRY *entry;
    PRIV_PTR priv;

    priv = sgui_get_priv(edtr->gui);

    /* Manage exp3 buttons which should be displayed and
     * unmanage exp3 buttons which should not be displayed
     * according to the current local force settings.
     */
    for (iloop = 0; iloop < edtr->n_menu; iloop++)
    {
        /* See if this is what we expect */
        if (NS_OBJECT_ID_NULL(edtr->menu[iloop].hhour_id) ||
            !(entry = po_get_object(edtr->db, &edtr->menu[iloop].hhour_id)) ||
            (PO_OBJECT_CLASS(entry) != objectClassHHour))
            continue;

        if (priv_get_local_force(priv, PO_HHOUR_DATA(entry).forceID))
        {
            XtManageChild(edtr->menu[iloop].button);
        }
        else
        {
            XtUnmanageChild(edtr->menu[iloop].button);
        }
    }
}

/* If an exp3 was being edited and it should be unmanaged due

```

```

    * to the local force setting, exit the editor.
    */
    if (!NS_OBJECT_ID_NULL(edtr->being_edited) &&
        (entry = po_get_object(edtr->db, &edtr->being_edited)) &&
        (PO_OBJECT_CLASS(entry) == objectClassHHour) &&
        (!priv_get_local_force(priv, PO_HHOUR_DATA(entry).forceID)))
    {
        sgui_leave_mode(edtr->gui);
    }
}

static void handle_local_force_event(
    int32 promoted_force,
    int32 display,
    EXP3_EDITOR*edtr)
{
    ForceID force = (ForceID) promoted_force;
    /* Make sure the exp3 buttons are managed/unmanaged according to
     * the local force setting.This is just a pass through routine.
     */
    manage_exp3_force(edtr);
}

static void expand_route_in_place (CTDB *ctdb,
    ROUTE_LIST *rlp)
{
    int32 n_verts;
    float64 *verts;
    ROUTE_LIST *wrp = rlp;
    int32 count, pnum, pdx;

    while (wrp)
    {
        if (wrp->sectype == ROUTE_TYPE_EMPTY)
        {
            printf ("expand_route: empty route encountered\n");
            break;
        }

        if (wrp->sectype == ROUTE_TYPE_POINTS)
        {
            wrp = wrp->next;
            continue;
        }

        /* Here if we actually have a road to expand */

        if (wrp->route.secroad.direction == ROUTE_FIRST_TO_LAST)
        {
            /*
             * If the previous element of the route list was a point,
             * then this is the road segment containing that point, and
             * first point on this segment should be dropped, since it
             * it is not actually on the route:
            */

```



```

static int32 BadGuyStatus = ApproachingBox;
if (tickNumber == 1)
{
    start_time = time_last_simulation_clock;
    elapsed_time = 0;
    system("play -v 8.1 /usr/share/sounds/gnibbles/pop.wav");
    NumberGoodVehicles = unitorg_get_hierarchy(good_guy_entry, FALSE,
    TRUE, vehicle_entry, 20);
    for (i=0; i < NumberGoodVehicles; i++)
    {
        GoodVehicleID[i] = vtab_vehicle_number(
            &PO_UNIT_DATA(vehicle_entry[i]).simulationID);
    }

    NumberBadVehicles = unitorg_get_hierarchy(bad_guy_entry, FALSE,
    TRUE, vehicle_entry, 20);
    for (i=0; i < NumberBadVehicles; i++)
    {
        BadVehicleID[i] = vtab_vehicle_number(
            &PO_UNIT_DATA(vehicle_entry[i]).simulationID);
    }

    for (i = 0; i < NumberGoodVehicles; i++)
    {
        fprintf(goodPos, "%dX %dY ", i, GoodVehicleID[i]);
    }
    fprintf(goodPos, "\n");
    for (i = 0; i < NumberBadVehicles; i++)
    {
        fprintf(badPos, "%dX %dY ", i, BadVehicleID[i]);
    }
    fprintf(badPos, "\n");
    tickNumber = tickNumber + 1;
}
else if (tickNumber == -99999999)
{
    printf("cant work yet \n");
}
else if (BadGuyStatus == LeavingBox)
{
    printf("EXP3 next one \n");
    system("play -v 8.1 /usr/share/sounds/gnibbles/pop.wav");
    BadGuyStatus = ApproachingBox;
    fclose(goodPos);
    fclose(badPos);
    system("play -v 8.1 /usr/share/sounds/gnibbles/pop.wav");
    system("play -v 8.1 /usr/share/sounds/gnibbles/pop.wav");
    sleep(1);
    unitutil_delete_object(db, &PO_OBJECT_ID(good_guy_entry), TRUE);
    unitutil_delete_object(db, &PO_OBJECT_ID(bad_guy_entry), TRUE);
    sched_deferred_fncl((SCHED_FUNCTION) exp3_tick, 100,
    my_group, A_PTR, db, A_END);
    tickNumber = -99999999;
}
else if (elapsed_time > 20*60*1000)
{

```

```

        /* MAF */
        unitutil_delete_object(db,&PO_OBJECT_ID(bad_guy_entry),TRUE);
        unitutil_delete_object(db,&PO_OBJECT_ID(good_guy_entry),TRUE);
        sched_deferred_fncl((SCHED_FUNCTION) exp3_tick,100,
        my_group,A_PTR,db,A_END);
        tickNumber = -99999999;
    }
else
{
    current_time = time_last_simulation_clock;
    elapsed_time = current_time - start_time;
    tickNumber = tickNumber + 1;
}
/* print out vehicle positions */
if (tickNumber > 0)
{
    fprintf(goodPos,"%d %ld",tickNumber,elapsed_time);
    for (i = 0; i < NumberGoodVehicles; i++)
    {
        ent_get_position_gcs(GoodVehicleID[i],pos);
        fprintf(goodPos,"%d %.3f %.3f ",GoodVehicleID[i],pos[X],pos[Y]);
    }
    fprintf(goodPos,"\n");
    fprintf(badPos,"%d %ld ",tickNumber,elapsed_time);
    for (i = 0; i < NumberBadVehicles; i++)
    {
        ent_get_position_gcs(BadVehicleID[i],pos);
        fprintf(badPos,"%d %.3f %.3f ",BadVehicleID[i],pos[X],pos[Y]);
    }
    fprintf(badPos,"\n");
}
    sched_deferred_fncl((SCHED_FUNCTION) exp3_schedule_tick,1000,my_group,A_PTR,db,A_END);
}

int exp3_get_number()
{
    int number;
    number = exp3_number;
    return number;
}

float64 exp3_get_start_time()
{
    float64 number;
    number = my_time;
    return number;
}

void make_box( float64 Cx,float64 Cy,
    float64 distX,float64 distY,
    float64 dX,float64 dY,
    float64 *x0,float64 *y0,
    float64 *x1,float64 *y1,
    float64 *x2,float64 *y2,
    float64 *x3,float64 *y3)
{

```

```

float64 ddx,ddy;
float64 xx0, yy0,xx1,yy1,xx2,yy2,xx3,yy3;

ddx = distX/2.0;
ddy = distY/2.0;

xx0 = Cx + (-1.0*ddx)*dX + (-1.0*ddy)*dY;
yy0 = Cy + (-1.0*ddx)*dY - (-1.0*ddy)*dX;

xx1 = Cx + (-1.0*ddx)*dX + (1.0*ddy)*dY;
yy1 = Cy + (-1.0*ddx)*dY - (1.0*ddy)*dX;

xx2 = Cx + (1.0*ddx)*dX + (1.0*ddy)*dY;
yy2 = Cy + (1.0*ddx)*dY - (1.0*ddy)*dX;

xx3 = Cx + (1.0*ddx)*dX + (-1.0*ddy)*dY;
yy3 = Cy + (1.0*ddx)*dY - (-1.0*ddy)*dX;

*x0 = xx0; *y0 = yy0; *x1 = xx1; *y1 = yy1;
*x2 = xx2; *y2 = yy2; *x3 = xx3; *y3 = yy3;
}

#define DEG_TO_RAD atan(1.0)/45.0
#define min_radius 5000
#define max_radius 10000

static int32 exp3_find_suitable_road(PO_DATABASE *db,
float64 XMin, float64 YMin,
float64 XMax, float64 YMax,
float64 *tx0,float64 *ty0,
float64 *tx1,float64 *ty1,
float64 *tx2,float64 *ty2,
float64 *tx3,float64 *ty3,
float64 *Obsx0,float64 *Obsy0,
float64 *Obsx1,float64 *Obsy1,
float64 *Obsx2,float64 *Obsy2,
float64 *Obsx3,float64 *Obsy3,
float64 *gMinX,float64 *gMinY,
float64 *gMaxX,float64 *gMaxY,
float64 *StartX,float64 *StartY,
float64 *StopX,float64 *StopY)
{
UnitClassunit;
float64 initialX,initialY,sx,sy,ex,ey;
float64 mx,my,Mx,My,ddx,ddy,borderx,bordery;
int32 found_start_point,found_end_point;
int32 trys,seg0,seg1,seg2,current_road,end_road_segment;
int32 subsec1,subsec2;
int32 point_in_the_box;
int32 i,k,Grade;
int32 attempts_to_find_road;
int32 max_points,observedRoadPoints,entranceRoadPoints,exitRoadPoints;
float64 adx,ady,Txx0, Tyy0,Txx1,Tyy1,Txx2,Tyy2,Txx3,Tyy3;
float64 Oxx0, Oyy0,Oxx1,Oyy1,Oxx2,Oyy2,Oxx3,Oyy3;
float64 MinX,MinY,MaxX,MaxY;
float64 ObsCx,ObsCy, MaxXRoad,MaxYRoad,MinXRoad,MinYRoad;

```

```

float64 TargCx,TargCy,len;
float64 TargBx,TargBy;
float64 initialDx,initialDy;
float64 road[2];
float64 radius,ccos,ssin,angle,px,py;
float64 MajorAxisLength, MinorAxisLength;
float64 ObserverMinorAxisLength;

observedRoadPoints = 50;
entranceRoadPoints = 30;
exitRoadPoints = 10;

attempts_to_find_road = 0;

Grade = UNACCEPTABLE;

while ( ( attempts_to_find_road < 50) && (Grade == UNACCEPTABLE))
{
    found_start_point = NO;
    trys = 0;
    /* 1a. find a starting point on the road */
    while ((found_start_point == NO)&& (trys < 1000))
    {
        trys = trys + 1;
        initialX = drand48()*(XMax - XMin) + XMin;
        initialY = drand48()*(YMax - YMin) + YMin;
        found_start_point = ctdb_nearest_road(ctdb,
        initialX,initialY,1000.0,&seg0,NULL);
        rt_find_point_on_road(ctdb,seg0,initialX,initialY,&subsec1, road);
        sx = road[0];mx = sx; Mx = sx;
        sy = road[1];my = sy; My = sy;
    }

    /* 1b. find the rest of the road */
    if (found_start_point == YES)
    {
        rt_find_point_on_road(ctdb,seg0,initialX,initialY,&subsec1, road);
        sx = road[0];mx = sx; Mx = sx;
        sy = road[1];my = sy; My = sy;
        angle = 2.0*PI*drand48();

        initialDx = sin(angle);
        initialDy = cos(angle);
        exp3_march_along_road(db,sx,sy,20.0,observedRoadPoints,
        initialDx,initialDy,
        observedRoadX,observedRoadY);
        current_road = seg0;
    }

    TargCx = 0.0; TargCy = 0.0;

    for (i = 0; i < observedRoadPoints; i++)
    {
        TargCx =TargCx+ observedRoadX[i];
        TargCy =TargCy+ observedRoadY[i];
        if (my > observedRoadY[i]) my = observedRoadY[i];
    }
}

```

```

        if (My < observedRoadY[i]) My = observedRoadY[i];
    }
    adx = observedRoadX[observedRoadPoints - 1] - observedRoadX[0];
    ady = observedRoadY[observedRoadPoints - 1] - observedRoadY[0];
    MajorAxisLength = sqrt(adx*adx + ady*ady) + 0.00001;
    adx = adx/MajorAxisLength;
    ady = ady/MajorAxisLength;
    MinorAxisLength = MajorAxisLength*2.0/3.0;

    TargCx =(observedRoadX[0] + observedRoadX[observedRoadPoints -1])/2.0;
    TargCy =(observedRoadY[0] + observedRoadY[observedRoadPoints -1])/2.0;

    ddx = adx*(observedRoadX[0] - TargCx) + ady*(observedRoadY[0] -TargCy);
    ddy = (My - my + 200.0);

    ObserverMinorAxisLength = 3500.0;

    ObsCx = TargCx+ 0.0*adx + (1.0*MinorAxisLength)*ady;
    ObsCy = TargCy+ 0.0*ady - (1.0*MinorAxisLength)*adx;

    /* 1c. make target box oriented on (adx,ady) (average road vector)*/
    make_box(TargCx,TargCy,MajorAxisLength,MinorAxisLength,adx,ady,
    &Txx0,&Tyy0,&Txx1,&Tyy1,&Txx2,&Tyy2,&Txx3,&Tyy3);

    /* 1d. try an obsever box*/
    make_box(ObsCx,ObsCy,MajorAxisLength,1.0*MinorAxisLength,adx,ady,
    &Oxx0,&Oyy0,&Oxx1,&Oyy1,&Oxx2,&Oyy2,&Oxx3,&Oyy3);
    if (look_for_rivers(Oxx0,Oyy0,Oxx1,Oyy1,Oxx2,Oyy2,Oxx3,Oyy3) == NO)
    {

        /* acceptable is 20% fully exposed 20% fully hidden */
        Grade = evaluate_los_surface(ctdb,
        Txx0,Tyy0,Txx1,Tyy1,Txx2,Tyy2,Txx3,Tyy3,
        Oxx0,Oyy0,Oxx1,Oyy1,Oxx2,Oyy2,Oxx3,Oyy3,
        adx,ady);
        if ( Grade == UNACCEPTABLE)
        {
            /* 1e. if one box doesn't work try the other*/
            ObsCx = TargCx + 0.0*adx - (1.0*MinorAxisLength)*ady;
            ObsCy = TargCy + 0.0*ady + (1.0*MinorAxisLength)*adx;
            make_box(ObsCx,ObsCy,MajorAxisLength,1.0*MinorAxisLength,
            -1.0*adx,-1.0*ady,&Oxx0, &Oyy0, &Oxx1, &Oyy1, &Oxx2,
            &Oyy2,&Oxx3,&Oyy3);

            if (look_for_rivers(Oxx0,Oyy0,Oxx1,Oyy1,Oxx2,Oyy2,Oxx3,Oyy3) == NO)
            {
                Grade = evaluate_los_surface(ctdb,
                Txx0,Tyy0,Txx1,Tyy1,Txx2,Tyy2,Txx3,Tyy3,
                Oxx0,Oyy0,Oxx1,Oyy1,Oxx2,Oyy2,Oxx3,Oyy3,
                adx,ady);
            }
        }
    }
}

```

```

*StartX = sx; *StartY = sy;
*StopX = ex; *StopY = ey;
if ( Grade == ACCEPTABLE)
{
    initialDx = observedRoadX[0] - observedRoadX[1];
    initialDy = observedRoadY[0] - observedRoadY[1];
    len = sqrt(initialDx*initialDx + initialDy*initialDy) + 0.0000001;
    initialDx = initialDx/len;
    initialDy = initialDy/len;
    exp3_march_along_road(db,sx,sy,75.0,entranceRoadPoints,
    initialDx,initialDy,
    entranceRoadX,entranceRoadY);
    k = 0;
    make_line(db,entranceRoadPoints,entranceRoadX,entranceRoadY,OCYellow);
    for (i = 0; i < entranceRoadPoints; i++)
    {
        my_roadX[k] = entranceRoadX[entranceRoadPoints - 1 - i];
        my_roadY[k] = entranceRoadY[entranceRoadPoints - 1 - i];
        k = k + 1;
    }
    make_line(db,observedRoadPoints,observedRoadX,observedRoadY,OCGreen);
    for (i = 0; i < observedRoadPoints; i++)
    {
        my_roadX[k] = observedRoadX[i];
        my_roadY[k] = observedRoadY[i];
        k = k + 1;
    }
    initialDx = observedRoadX[observedRoadPoints - 1] -
    observedRoadX[observedRoadPoints - 2];
    initialDy = observedRoadY[observedRoadPoints - 1] -
    observedRoadY[observedRoadPoints - 2];
    len = sqrt(initialDx*initialDx + initialDy*initialDy) + 0.0000001;
    initialDx = initialDx/len;
    initialDy = initialDy/len;
    exp3_march_along_road(db,observedRoadX[observedRoadPoints - 1],
    observedRoadY[observedRoadPoints - 1],
    50.0,exitRoadPoints,
    initialDx,initialDy,
    exitRoadX,exitRoadY);
    make_line(db,exitRoadPoints,exitRoadX,exitRoadY,OCBlack);
    for (i = 0; i < observedRoadPoints; i++)
    {
        my_roadX[k] = exitRoadX[i];
        my_roadY[k] = exitRoadY[i];
        k = k + 1;
    }
    max_points = entranceRoadPoints + observedRoadPoints + exitRoadPoints;
    *StartX = my_roadX[0]; *StartY = my_roadY[0];
    *StopX = my_roadX[max_points - 1]; *StopY = my_roadX[max_points - 1];
    BadGuyRouteEntry =
        make_line(db,max_points,my_roadX,my_roadY,OCBlue);
}
MinX = Txx0; MinY = Tyy0; MaxX = Txx0; MaxY = Tyy0;
if (MaxX < Txx1) MaxX = Txx1; if (MaxY < Tyy1) MaxY = Tyy1;
if (MaxX < Txx2) MaxX = Txx2; if (MaxY < Tyy2) MaxY = Tyy2;

```

```

        if (MaxX < Txx3) MaxX = Txx3; if (MaxY < Tyy3) MaxY = Tyy3;

        if (MaxX < Oxx0) MaxX = Oxx0; if (MaxY < Oyy0) MaxY = Oyy1;
        if (MaxX < Oxx1) MaxX = Oxx1; if (MaxY < Oyy1) MaxY = Oyy1;
        if (MaxX < Oxx2) MaxX = Oxx2; if (MaxY < Oyy2) MaxY = Oyy2;
        if (MaxX < Oxx3) MaxX = Oxx3; if (MaxY < Oyy3) MaxY = Oyy3;

        if (MinX > Txx1) MinX = Txx1; if (MinY > Tyy1) MinY = Tyy1;
        if (MinX > Txx2) MinX = Txx2; if (MinY > Tyy2) MinY = Tyy2;
        if (MinX > Txx3) MinX = Txx3; if (MinY > Tyy3) MinY = Tyy3;

        if (MinX > Oxx0) MinX = Oxx0; if (MinY > Oyy0) MinY = Oyy0;
        if (MinX > Oxx1) MinX = Oxx1; if (MinY > Oyy1) MinY = Oyy1;
        if (MinX > Oxx2) MinX = Oxx2; if (MinY > Oyy2) MinY = Oyy2;
        if (MinX > Oxx3) MinX = Oxx3; if (MinY > Oyy3) MinY = Oyy3;

        *tx0 = Txx0; *ty0 = Tyy0; *tx1 = Txx1; *ty1 = Tyy1;
        *tx2 = Txx2; *ty2 = Tyy2; *tx3 = Txx3; *ty3 = Tyy3;
        *Obsx0 = Oxx0; *Obsy0 = Oyy0; *Obsx1 = Oxx1; *Obsy1 = Oyy1;
        *Obsx2 = Oxx2; *Obsy2 = Oyy2; *Obsx3 = Oxx3; *Obsy3 = Oyy3;

        *gMinX = MinX; *gMinY = MinY; *gMaxX = MaxX; *gMaxY = MaxY;

        attempts_to_find_road = attempts_to_find_road+ 1;
    }
}
return Grade;
}

#define PI 4.0*atan(1.0)

void exp3_march_along_road(PO_DATABASE *db,float64 sx,float64 sy,float64 step,
    int32 max_points,
    float64 initialDx,float64 initialDy,
    float64 *CurrentRoadX,float64 *CurrentRoadY)
{
    int32 current_point,my_point;
    int32 subsec1,subsec2,new_road;
    int32 found_it,trys,seg0,seg1,seg2,current_road,end_road_segment;
    int32 ii,k_step;
    float64 px,py,ax,ay;
    float64 my_dx,my_dy,my_slope,my_dist,my_dist2;
    float64 road[2];
    float64 angle,ex,ey,SmallerStep;
    float64 step_end,ddx,ddy,temp_dist;

    CurrentRoadX[0] = sx;
    CurrentRoadY[0] = sy;
    angle = (float) degrees/180.0*PI;
    ii = 1;
    ax = 0.0; ay = 0.0;
    while (ii< max_points)
    {
        if ( ii == 1)
        {
            /* vector form start to endpoint */

```

```

my_dx = initialDx;
my_dy = initialDy;

px = sx + my_dx;
py = sy + my_dy;

/* pick the closest road point to that one */
found_it = ctdb_nearest_road(ctdb,px,py,100.0,&seg1, NULL);
current_road = seg1;
rt_find_point_on_road(ctdb,seg1,px,py,&subsec2, road);
CurrentRoadX[ii] = road[0];
CurrentRoadY[ii] = road[1];
ii = ii + 1;
}
else if ( ii > 1)
{
/* vector along the road between last two points */
my_dx = (CurrentRoadX[ii-1] - CurrentRoadX[ii-2]);
my_dy = (CurrentRoadY[ii-1] - CurrentRoadY[ii-2]);
my_dist = sqrt( my_dx*my_dx + my_dy*my_dy);
if (my_dist < 0.00001)
{
my_dx = initialDx;
my_dy = initialDy;
}
ddx = my_dx/(my_dist + 0.00001)*step;
ddy = my_dy/(my_dist + 0.00001)*step;

px = CurrentRoadX[ii-1] + ddx;
py = CurrentRoadY[ii-1] + ddy;

/* pick the closest road point to that one */
found_it = ctdb_nearest_road(ctdb,px,py,step,&seg1, NULL);
if (current_road != seg1)
{
/* save this point */
rt_find_point_on_road(ctdb,seg1,px,py,&subsec2, road);
ax = road[0];
ay = road[1];
new_road = seg1;

printf("EXP3 need to cross an intersection at the %dth point \n",ii);
k_step = 99;
seg2 = seg1;
while((seg2 != current_road) && (k_step > 0))
{
SmallerStep = step/100.0*(float) k_step;
ddx = my_dx/(my_dist + 0.00001)*SmallerStep;
ddy = my_dy/(my_dist + 0.00001)*SmallerStep;
px = CurrentRoadX[ii-1] + ddx;
py = CurrentRoadY[ii-1] + ddy;
found_it = ctdb_nearest_road(ctdb,px,py,20.0,&seg2, NULL);
k_step = k_step - 1;
}
/* add the intersectionpoint to the road */
rt_find_point_on_road(ctdb,seg1,px,py,&subsec2, road);

```

```

        CurrentRoadX[ii] = road[0];
        CurrentRoadY[ii] = road[1];
        ii = ii + 1;
        /* cross the intersection */
        CurrentRoadX[ii] = ax;
        CurrentRoadY[ii] = ay;
        ii = ii + 1;
        current_road = new_road;
    }
    else
    {
        rt_find_point_on_road(ctdb,seg1,px,py,&subsec2, road);
        CurrentRoadX[ii] = road[0];
        CurrentRoadY[ii] = road[1];
        ii = ii + 1;
        current_road = seg1;
    }
}
}

/*
I want my overly on the main screen.Routines copied from elsewhere in the OneSAF code
*/

static void find_an_overlay(
    PO_DB_ENTRY *entry,
    struct overlay_search *search)
{
    /* Make sure the overlay is okay to display. */
    if (!search->found &&
        (PO_OBJECT_CLASS(entry) == objectClassOverlay) &&
        !PO_OVERLAY_DATA(entry).working &&
        !PO_OVERLAY_DATA(entry).scratch)
    {
        search->found = 1;
        *search->id = PO_OBJECT_ID(entry);
        search->entry = entry;
    }
}

static void get_an_overlay(
    PO_DATABASE *db,
    ObjectID*id)
{
    PO_DB_ENTRY *entry;
    struct overlay_search search;
    PO_BUFFER(OverlayClass, variant, ovl_buf);

    /* See if we can find an overlay, *any* displayable overlay */
    search.found = 0;
    search.id = id;
    po_query_for_current_class(db, objectClassOverlay,
        (PO_QUERY_HANDLER)find_an_overlay,
        (PO_USER_DATA_TYPE)&search);

    if (search.found)

```

```

return;

bzero(ovl_buf, sizeof(ovl_buf));
variant->color = OCBlack;
variant->forceID = forceIDIrrelevant;
strcpy((char *)variant->name, "<Unnamed>");
if (!(entry = po_create_object(db, NULL, objectClassOverlay, FALSE,
    variant, PRO_PO_OVERLAY_CLASS_SIZE,
    NULL)))
{
    printf("EXP3 Error creating default overlay in PO database: %s\n",
        po_errlist[po_errno]);
}
*id = PO_OBJECT_ID(entry);
}

void draw_box( PO_DATABASE *db,int32 color,int32 number,
    float64 x0,float64 y0,
    float64 x1,float64 y1,
    float64 x2,float64 y2,
    float64 x3,float64 y3)
{
    int32 ic;
    PO_BUFFER(LineClass, my_line, buf);
    PO_BUFFER(TextClass, my_text, buf2);
    PO_DB_ENTRY*my_line_entry,*my_text_entry;
    char exerciseNumber[20];

    get_an_overlay(db, &my_line->overlayID);
    my_line->width = 2;
    my_line->closed = FALSE;
    my_line->splined = FALSE;
    my_line->route = FALSE;
    my_line->style = LSplain;
    my_line->dashed = True;
    my_line->beginArrowHead = noArrowHead;
    my_line->endArrowHead = noArrowHead;
    my_line->thickness = 2;
    my_line->color = color;
    my_line->pointCount = 5;
    my_line->methodology = SAFMethodFundamental;
    my_line->shouldBeSimulated = TRUE;

    my_line->points[0].pointNumber = 0;
    my_line->points[0].variant.location.x = x0;
    my_line->points[0].variant.location.y = y0;
    my_line->points[0].pointType = PTLocation;
    my_line->points[0].variant.location.cell = GCS_ILLEGAL_CELL;

    my_line->points[1].pointNumber = 1;
    my_line->points[1].variant.location.x = x1;
    my_line->points[1].variant.location.y = y1;
    my_line->points[1].pointType = PTLocation;
    my_line->points[1].variant.location.cell = GCS_ILLEGAL_CELL;

```

```

my_line->points[2].pointNumber = 2;
my_line->points[2].variant.location.x = x2;
my_line->points[2].variant.location.y = y2;
my_line->points[2].pointType = PTLocation;
my_line->points[2].variant.location.cell = GCS_ILLEGAL_CELL;

my_line->points[3].pointNumber = 3;
my_line->points[3].variant.location.x = x3;
my_line->points[3].variant.location.y = y3;
my_line->points[3].pointType = PTLocation;
my_line->points[3].variant.location.cell = GCS_ILLEGAL_CELL;

my_line->points[4].pointNumber = 4;
my_line->points[4].variant.location.x = x0;
my_line->points[4].variant.location.y = y0;
my_line->points[4].pointType = PTLocation;
my_line->points[4].variant.location.cell = GCS_ILLEGAL_CELL;

my_line_entry = po_create_object(db, NULL, objectClassLine,
FALSE, my_line,
PRO_PO_LINE_CLASS_SIZE(my_line->pointCount), NULL);
sprintf(my_text->text, "Exp-%d ", number);
get_an_overlay(db, &my_text->overlayID);
my_text->color = OCBlack;
my_text->location.x = (x0 + x1 + x2 + x3)/4.0;
my_text->location.y = (y0 + y1 + y2 + y3)/4.0;
my_text->length = 10;
my_text_entry = po_create_object(db, NULL, objectClassText,
FALSE, my_text,
PRO_PO_TEXT_CLASS_SIZE(my_text->length), NULL);
}

static PO_DB_ENTRY *make_point( PO_DATABASE *db,
ObjectID *overlay,
float64 x, float64 y,
float64 direction)
{
PO_DB_ENTRY *my_point_entry;
PO_BUFFER(PointClass, my_point, buf);
my_point->overlayID = *overlay;
my_point->style = PSTRP;
my_point->color = OCYellow;
my_point->dashed = FALSE;
my_point->location.x = x;
my_point->location.y = y;
my_point->direction = direction;
sprintf(my_point->text, " ");
my_point_entry = po_create_object(db, NULL, objectClassPoint,
FALSE, my_point, sizeof(my_point), NULL);
return my_point_entry;
}

static PO_DB_ENTRY *make_line( PO_DATABASE *db, int numPoints,
float64 *x, float64 *y,
int32 color)
{

```

```

int32 i;
PO_BUFFER(LineClass, my_line, buf);
PO_DB_ENTRY*my_line_entry;

get_an_overlay(db, &my_line->overlayID);
my_line->width = 3;
my_line->closed = FALSE;
my_line->splined = FALSE;
my_line->route = FALSE;
my_line->style = LSplain;
my_line->dashed = TRUE;
my_line->beginArrowHead = noArrowHead;
my_line->endArrowHead = noArrowHead;
my_line->thickness = 5;
if (color == OCBlue) my_line->thickness = 1;
my_line->color = color;
my_line->pointCount = numPoints;
my_line->methodology = SAFMethodFundamental;
my_line->shouldBeSimulated = TRUE;
for (i = 0; i < numPoints; i++)
{
    my_line->points[i].pointNumber = 0;
    my_line->points[i].variant.location.x = x[i];
    my_line->points[i].variant.location.y = y[i];
    my_line->points[i].pointType = PTLocation;
    my_line->points[i].variant.location.cell = GCS_ILLEGAL_CELL;
}
my_line_entry = po_create_object(db, NULL, objectClassLine,
    FALSE, my_line,
    PRO_PO_LINE_CLASS_SIZE(my_line->pointCount), NULL);
}

int32 exp3_time_to_quit(PO_DATABASE *db)
{
    float64 pos[XYZC],XX,YY,xxx,yyy,dist;
    int32 lengthCheck,widthCheck;
    int32 i,ans;
    ans = NO;
    for (i = 0; (i < NumberBadVehicles && (ans == NO)); i++)
    {
        ent_get_position_gcs(BadVehicleID[i],pos);
        /* translate */
        xxx = pos[X] - tx0;
        yyy = pos[Y] - ty0;
        /* rotate */
        XX = tdx*xxx + tdy*yyy;
        YY = tdy*xxx - tdx*yyy;
        if (( XX < 0 ) || (YY < 0))
        {
            return NO;
        }
        else if (( XX < twidth) && (YY < tlength))
        {
            printf("EXP3 inside box \n");
        }
        else
    }
}

```

```

        {
            printf("EXP3 outside the box \n");
            ans = YES;
            return ans;
        }
    }
    return ans;
}

#define SCALE 25

void dump_terrain_surface( char *file_name,
    float64 MinX,float64 MinY,
    float64 MaxX,float64 MaxY)
{
    int32 ix,iy,numX,numY;
    float XX,YY,ZZ;
    FILE *myfile;
    char elevations[20];
    sprintf(elevations,"%sElev.%d",file_name,exp3_count);
    myfile = fopen(elevations,"w");
    numX = (MaxX - MinX)/SCALE;
    numY = (MaxY - MinY)/SCALE;
    for (iy = 0; iy < numY; iy++)
    {
        YY = MinY + (float) iy*SCALE;
        for (ix = 0; ix < numX; ix++)
        {
            XX = MinX + (float) ix*SCALE;
            ZZ = ctdb_lookup_elevation(ctdb, XX,YY);
            fprintf(myfile,"%%.3lf %%.3lf %%.3lf \n", XX,YY,ZZ);
        }
    }
    fclose(myfile);
}

#ifdef USE_MOTIF
#else
#endif /*USE_MOTIF*/

```

NO. OF
COPIES ORGANIZATION

1 DEFENSE TECHNICAL
(PDF INFORMATION CTR
ONLY) DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FT BELVOIR VA 22060-6218

1 COMMANDING GENERAL
US ARMY MATERIEL CMD
AMCRDA TF
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 US MILITARY ACADEMY
MATH SCI CTR EXCELLENCE
MADN MATH
THAYER HALL
WEST POINT NY 10996-1786

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE AD IM DR
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

3 DIR USARL
AMSRD ARL SE RM
E BUKE
G GOLDMAN
AMSRD ARL SE DC
A GOLDBERG
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DOD JOINT CHIEF OF STAFF
J39 CAPABILITIES DIV
J M BROWNELL
THE PENTAGON
RM 2C865
WASHINGTON DC 20301

1 DIR CIA
D MOORE
WASHINGTON DC 20505-0001

3 PM ABRAMS TANK SYSTEM
SFAE GCS AB
KOTCHMAN
P LEITHEISER
H PETERSON
WARREN MI 48397-5000

1 PM M1A2
SFAE GCS AB
R LOVETT
WARREN MI 48397-5000

1 PM M1A1
SFAE GCS AB
L C MILLER JR
WARREN MI 48397-5000

1 PM BFVS
SFAE GCS BV
C L MCCOY
WARREN MI 48397-5000

1 PM BFVS
ATZB BV
C BETEK
FT BENNING GA 31905

1 PM M2/M3 BFVS
SFAE GCS BV
J MCGUINESS
WARREN MI 48397-5000

NO. OF
COPIES ORGANIZATION

3 PM BCT
SFAE GCS BCT
R D OGG JR
J GERLACH
T DEAN
WARREN MI 48397-5000

1 PM IAV
SFAE GCS BCT
J PARKER
WARREN MI 48397-5000

1 PM NIGHT VISION/RSTA
SFAE IEW&S NV
BOWMAN
10221 BURBECK RD
FT BELVOIR VA 22060-5806

1 NIGHT VISION & ELECTRONIC
SENSORS DIRECTORATE
A F MILTON
10221 BURBECK RD
STE 430
FT BELVOIR VA 22060-5806

1 CDR
US ARMY TRADOC
ATINZA
R REUSS
BLDG 133
FT MONROE VA 23651

1 CDR
US ARMY TRADOC
ATINI
C GREEN
BLDG 133
FT MONROE VA 23651

1 OFC OF THE SECY OF DEFENSE
CTR FOR COUNTERMEASURES
M A SCHUCK
WSMR NM 88002-5519

1 US SOCOM
SOIO JA F
J GOODE
7701 TAMPA POINT BLVD
BLDG 501
MCDILL AFB FL 33621-5323

NO. OF
COPIES ORGANIZATION

1 CDR
US ARMY ARMOR CTR & FT KNOX
TSM/ABRAMS
D SZYDLOSKI
FT KNOX KY 40121

1 CDR
US AMBL
J JUGHES
FT KNOX KY 40121

1 DIR OF COMBAT DEVELOPMENT
ATZK FD W MEINSHAUSEN
BLDG 1002 RM 326
1ST CAVALRY DIV RD
FT KNOX KY 40121-9142

1 CMDG OFFICER
MARINE CORPS INTEL ACTIVITY
W BARTH
3300 RUSSELL RD STE 250
QUANTICO VA 22134-5011

1 UNIV OF SOUTH FL
COMPUTER SCIENCE AND ENGRG
R MURPHY
4202 E FOWLER AVE ENB342
TAMPA FL 33620-5399

1 APPLIED PHYSICS LAB
T NEIGHOFF
11100 JOHNS HOPKINS RD
LAUREL MD 20723-6099

1 SAIC
K A JAMISON
PO BOX 4216
FT WALTON BEACH FL 32549

23 CDR
USA TACOM ARDEC
AMSTA AR TD
M DEVINE
M FISETTE
AMSTA AR FSA S
R KOPMANN
H KERWIEN
K JONES
A FRANCHINO
AMSTA AR FSA P
D PASCUA
AMSTA AR FSA M
J FENECK

NO. OF
COPIES ORGANIZATION

AMSTA AR FSP
D LADD
M CILLI
AMSTA AR CCH A
M PALTHINGAL
A VELLA
E LOGSDON
R CARR
M MICOLICH
M YOUNG
AMSTA AR QAC
R SCHUBERT
AMSTA AR FSP G
A PEZZANO
R SHORR
AMSTA AR FSA T
A LAGASCA
AMSTA AR FSP I
R COLLETT
AMSTA AR WE C
R FONG
S TANG
PICATINNY ARSENAL NJ 07806-5000

5 PEO PM MORTAR SYSTEMS
SFAE AMO CAS IFM
L BICKLEY
M SERBAN
K SLIVOVSKY
SFAE GCS TMA
R KOWALSKI
SFAE GCS TMA PA
E KOPACZ
PICATINNY ARSENAL NJ 07860-5000

8 PEO GCS
SFAE GCS
C GAGNON
SFAE GCS W
A PUZZUOLI
SFAE GCS BV
J PHILLIPS
SFAE GCS LAV
T LYTLE
SFAE GCS AB SW
PATTISON
SFAE GCS AB LF
PAULSON
SFAE GCS LAV M
T KLER
SFAE GCS LAV FCS
ASOKLIS
WARREN MI 48397-5000

NO. OF
COPIES ORGANIZATION

23 CDR US ARMY TACOM
AMSTA TR
R MCCLELLAND
BAGWELL
AMSTA TR R
J PARKS
S SCHEHR
D THOMAS
C ACIR
J SOLTESZ
S CAITO
K LIM
J REVELLO
B BEAUDOIN
B RATHGEB
M CHAIT
S BARSHAW
AMSTA CM XSF
R DRITLEIN
HENDERSON
HUTCHINSON
SCHWARZ
PATHAK
R HALLE
J ARKAS
G SIMON
AMSTA TA
J CHAPIN
WARREN MI 48397-5000

3 MIT LINCOLN LAB
J HERD
G TITI
D ENGREN
244 WOOD ST
LEXINGTON MA 02420-9108

2 THE UNIV OF TEXAS AT AUSTIN
INSTIT FOR ADVANCED TECH
PO BOX 20797
I MCNAB
S BLESS
AUSTIN TX 78720-2797

1 INNOVATIVE SURVIVABILITY TECH
J STEEN
PO BOX 1989
GOLETA CA 93116

1 SUNY BUFFALO
ELECTRICAL ENGRG DEPT
J SARJEANT
PO BOX 601900
BUFFALO NY 14260-1900

NO. OF
COPIES ORGANIZATION

1 GENERAL DYNAMICS
LAND SYSTEMS
D GERSDORFF
PO BOX 2074
WARREN MI 49090-2074

1 CDR US ARMY CECOM
W DEVILBISS
BLDG 600
FT MONMOUTH NJ 07703-5206

1 MARCORSYSCOM/CBG
J DOUGLAS
QUANTICO VA 22134-5010

2 COMMANDER
USAIC
ATZB CDF
J LANE
D HANCOCK
FT BENNING GA 31905

1 DIR USARL
AMSRD ARL SL EA
R CUNDIFF
AMSRL SL EM
J THOMPSON
WSMR NM 88001-5513

4 UNITED DEFENSE ADV DEV CTR
K GROVES
J FAUL
T WINANT
V HORVATICH
328 BROKAW RD
SANTA CLARA CA 95050

2 NORTHROP GRUMMAN CORP
A SHREKENHAMER
D EWART
1100 W HOLLYVALE ST
AZUSA CA 91702

1 CDR US ARMY AMCOM
AMSAM RD ST WF
D LOVELACE
REDSTONE ARSENAL AL 35898-5247

1 OFC OF THE SECY OF DEFNS
ODDRE (R&T) G SINGLEY
THE PENTAGON
WASHINGTON DC 20301-3080

NO. OF
COPIES ORGANIZATION

1 OSD
OUSD (A&T) ODDR&E
J R TREW
THE PENTAGON
WASHINGTON DC 20310-0560

1 US MILITARY ACADEMY
MATHEMATICAL SCIENCES
CTR OF EXCELLENCE
DEPT OF MATHEMATICAL SCIENCES
MDN A
M D PHILLIPS
THAYER HALL
WEST POINT NY 10996-1786

2 DIR
US ARMY WATERWAYS
EXPER STATION
R AHLVIN
3909 HALLS FERRY RD
VICKSBURG MS 39180-6199

2 NATL INST STAN AND TECH
K MURPHY
100 BUREAU DR
GAITHERSBURG MD 20899

3 CDR
US ARMY MMBL
J BURNS
BLDG 2021
BLACKHORSE REGIMENT DR
FT KNOX KY 40121

2 DIR
NASA JET PROPULSION LAB
L MATHIES
K OWENS
4800 OAK GROVE DR
PASADENA CA 91109

1 DIR
AMCOM MRDEC
AMSMI RD
W MCCORKLE
REDSTONE ARSENAL AL 35898-5240

1 CDR CECOM
SP & TERRESTRIAL COM DIV
AMSEL RD ST MC
M SOICHER
FT MONMOUTH NJ 07703-5203

NO. OF
COPIES ORGANIZATION

1 CDR
US ARMY INFO SYS ENGRG CMD
ASQB OTD
F JENIA
FT HUACHUCA AZ 85613-5300

1 CDR
US ARMY NATICK RDEC
ACTING TECHNICAL DIR
SSCNC
T BRANDLER
NATICK MA 01760-5002

1 CDR
ARMY RESEARCH OFC
4300 S MIAMI BLVD
RESEARCH TRIANGLE PARK NC
27709

1 CDR
US ARMY STRICOM
J STAHL
12350 RESEARCH PKWY
ORLANDO FL 32826-3726

1 CDR
US ARMY TRADOC
BATTLE LAB INTEGRATION 7
TECH DIR
ATCD B J
A KLEVECZ
FT MONROE VA 23651-5850

ABERDEEN PROVING GROUND

1 PM ODS
SFAE CBD
B WELCH
BLDG 4475
APG MD 21010-5424

1 CDR US ARMY ABERDEEN
TEST CENTER
VPG TEAM
J CORDE LANE PHD
VIRTUAL PROVING GROUND TEAM
ABERDEEN TEST CENTER
400 COLLERAN RD BLD 321

1 CDR
US ARMY EDGEWOOD RDEC
SCBRD TD J VERVIER
APG MD 21010-5000

NO. OF COPIES	ORGANIZATION
2	US ARMY TECOM AMSTE CD B SIMMONS AMSTE CD M R COZBY APG MD 21005-5000
3	DIR US AMSAA AMXS D M MCCARTHY B SIEGEL P TOPPER APG MD 21005-5067
4	CDR USA ATC STEACCO ELLIS STEAC TD J FASIG STEAC TE H CUNNINGHAM STEAC RM C A MOORE STEAC TE F P OXENBERG A SCRAMLIN APG MD 21005-5000
32	DIR USARL AMSRD ARL WM J SMITH E SCHMIDT B RINGER T ROSENBERGER B BURNS C SHOEMAKER J BORNSTEIN AMSRD ARL WM B W CIEPIELLA AMSRD ARL WM BA D LYONS AMSRD ARL WM BD B FORCH AMSRD ARL WM MB L BURTON AMSRD ARL WM BC P PLOSTINS

NO. OF COPIES	ORGANIZATION
	AMSRD ARL WM TE G THOMSON T KOTTKE M MCNEIR P BERNING J POWELL C HUMMER AMSRD ARL WM TC R COATES AMSRD ARL SL BG M ENDERLEIN AMSRD ARL SL EM C GARRETT AMSRD ARL WM BF D WILKERSON J LACETERA T HAUG M FIELDS W OBERLE J WALD J WALL M BARANOSKI W OBERLE C PATTERSON P BUTLER R ANDERSON G SAUERBORN